

项目实战分册

微视频精编版

明日科技 编著

SQL Server 从入门到精通



71集高清微课视频，随时可学
217个应用实例
6个企业项目案例

在玩中嗨翻SQL Server
趣味解读 / 易学易用 / 学练结合
知识点讲解+实例+项目

海量资源，可查可练
- 实例资源库
- 模块资源库
- 项目资源库
- 测试题库系统

清华大学出版社

软件开发微视频讲堂

SQL Server从入门到精通

（微视频精编版）

明日科技　编著

清華大學出版社
北　京

内 容 简 介

本书内容浅显易懂，实例丰富，详细介绍了从基础入门到 SQL Server 数据库高手需要掌握的知识。

全书分为上下两册：核心技术分册和项目实战分册。核心技术分册共 2 篇 19 章，包括数据库基础、SQL Server 2014 安装与配置、创建和管理数据库、操作数据表、操作表数据、SQL 函数的使用、视图操作、Transact-SQL 语法基础、数据的查询、子查询与嵌套查询、索引与数据完整性、流程控制、存储过程、触发器、游标的使用、SQL 中的事务、SQL Server 高级开发、SQL Server 安全管理和 SQL Server 维护管理等内容。项目实战分册共 6 章，运用软件工程的设计思想，介绍了腾宇超市管理系统、学生成绩管理系统、图书商城、房屋中介管理系统、客房管理系统和在线考试系统共 6 个完整企业项目的真实开发流程。

本书除纸质内容外，配书资源包中还给出了海量开发资源，主要内容如下。

- ☑ **微课视频讲解：总时长 8 小时，共 71 集**
- ☑ **实例资源库：126 个实例及源码分析**
- ☑ **模块资源库：15 个经典模块完整展现**
- ☑ **项目资源库：15 个企业项目开发过程**
- ☑ **测试题库系统：596 道能力测试题目**

本书适合有志于从事软件开发的初学者、高校计算机相关专业学生和毕业生，也可作为软件开发人员的参考手册，或者高校的教学参考书。

图书在版编目（CIP）数据

SQL Server 从入门到精通：微视频精编版 / 明日科技编著. —北京：清华大学出版社，2020.7
（软件开发微视频讲堂）
ISBN 978-7-302-52090-0

Ⅰ. ①S… Ⅱ. ①明… Ⅲ. ①关系数据库系统 Ⅳ. ①TP311.132.3

中国版本图书馆 CIP 数据核字（2019）第 010403 号

责任编辑：贾小红
封面设计：魏润滋
版式设计：文森时代
责任校对：马军令
责任印制：杨　艳

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：203mm×260mm　　**印　　张**：38　　**字　　数**：1025 千字
版　　次：2020 年 9 月第 1 版　　**印　　次**：2020 年 9 月第 1 次印刷
定　　价：99.80 元（全 2 册）

产品编号：079180-01

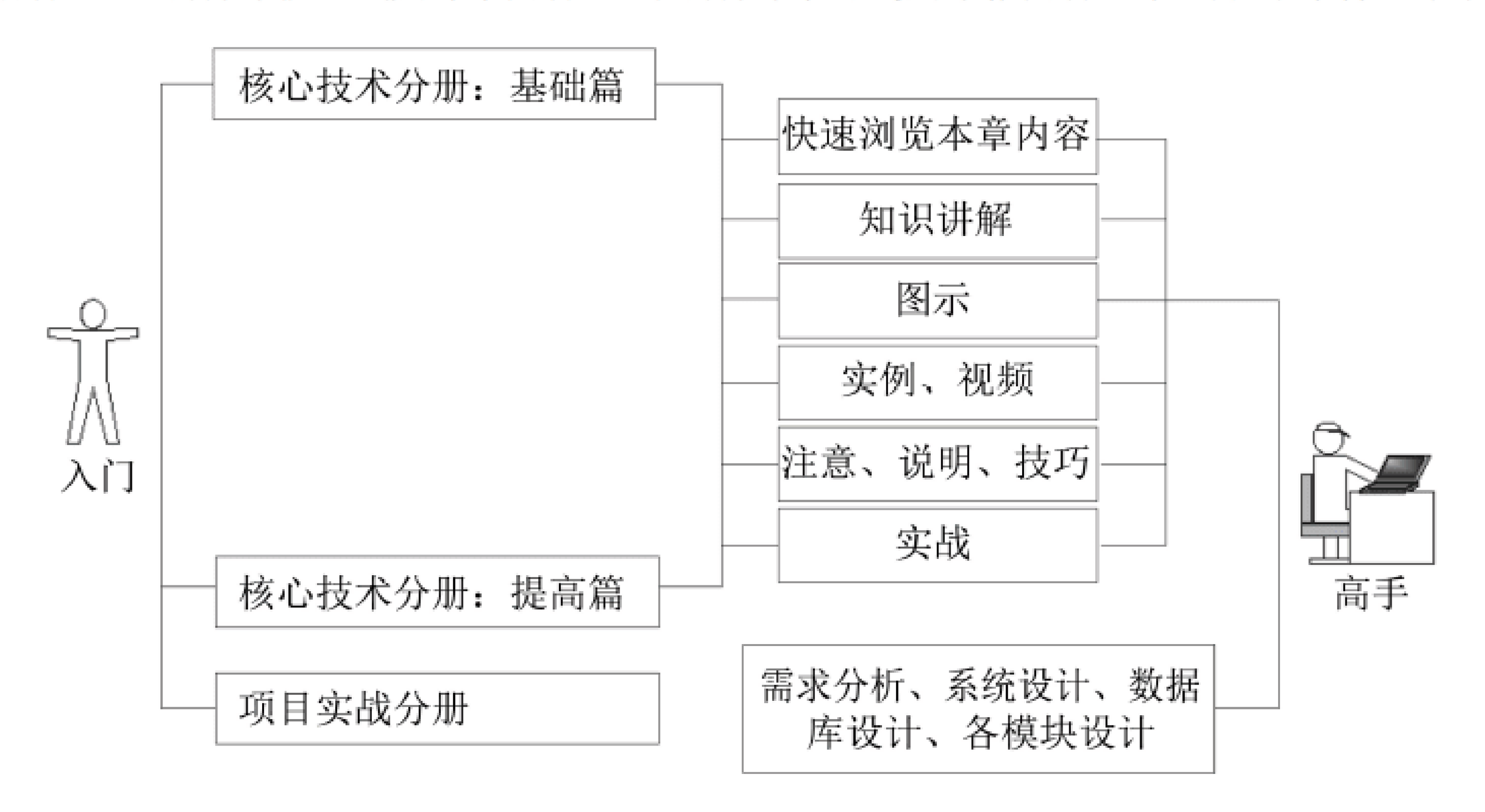
前　言

Preface

SQL Server 是由美国微软（Microsoft）公司制作并发布的一种性能优越的关系型数据库管理系统（Relational Database Management System，RDBMS），因其具有良好的数据库设计、管理与网络功能，又与 Windows 系统紧密集成，因此成为数据库产品的首选。

本书内容

本书分上下两册，上册为核心技术分册，下册为项目实战分册，大体结构如下图所示。



核心技术分册共分 2 篇 19 章，提供了从基础入门到 SQL Server 数据库高手所必备的各类知识。

基础篇：介绍了数据库基础、SQL Server 2014 安装与配置、创建和管理数据库、操作数据表、操作表数据、SQL 函数的使用、视图操作、Transact-SQL 语法基础、数据的查询、子查询与嵌套查询等内容，并结合大量的图示、实例、视频和实战等，使读者快速掌握 SQL 语言基础。

提高篇：介绍了索引与数据完整性、流程控制、存储过程、触发器、游标的使用、SQL 中的事务、SQL Server 高级开发、SQL Server 安全管理和 SQL Server 维护管理等内容。学习完本篇，能够掌握比较高级的 SQL 及 SQL Server 管理知识，并对数据库进行管理。

项目实战分册共 6 章，运用软件工程的设计思想，介绍了 6 个完整企业项目（腾宇超市管理系统、学生成绩管理系统、图书商城、房屋中介管理系统、客房管理系统和在线考试系统）的真实开发流程。书中按照“需求分析→系统设计→数据库设计→项目主要功能模块的实现”的流程进行介绍，带领读者亲身体验开发项目的全过程，提升实战能力，实现从小白到高手的跨越。

本书特点

☑　**由浅入深，循序渐进**。本书以初、中级读者为对象，先从 SQL 语言基础学起，再学习数据库

对象的使用，如视图、存储过程、触发器等，最后学习开发一个完整项目。讲解过程中步骤详尽，版式新颖，使读者在阅读时一目了然，从而快速掌握书中内容。

☑ **实例典型，轻松易学**。通过例子学习是最好的学习方式，本书通过“一个知识点、一个例子、一个结果、一段评析，一个综合应用”的模式，透彻详尽地讲述了实际开发中所需的各类知识。另外，为了便于读者阅读程序代码，快速学习编程技能，书中绝大多数代码提供了注释。

☑ **微课视频，讲解详尽**。本书为便于读者直观感受程序开发的全过程，书中大部分章节都配备了教学微视频，使用手机扫描正文小节标题一侧的二维码，即可观看学习，能快速引导初学者入门，感受编程的快乐和成就感，进一步增强学习的信心。

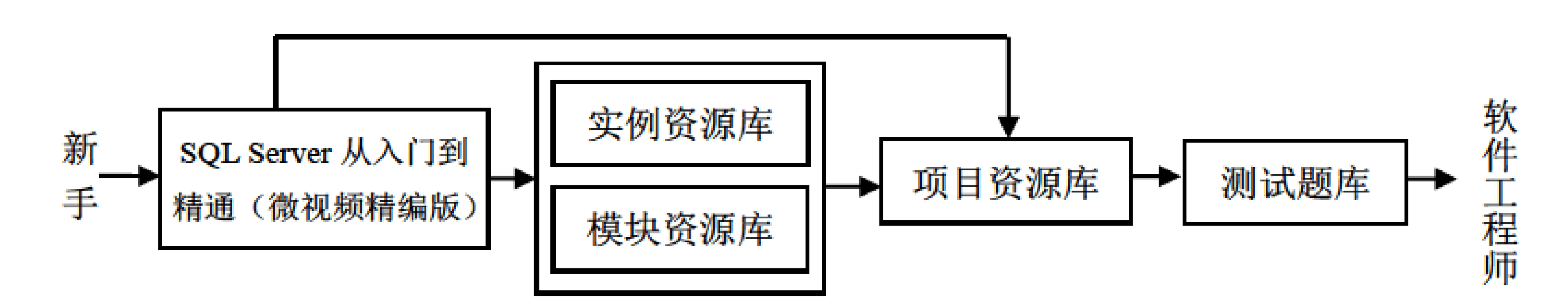
☑ **精彩栏目，贴心提醒**。本书根据需要在各章安排了“注意”“说明”等小栏目，让读者可以在学习过程中更轻松地理解相关知识点及概念，更快地掌握个别技术的应用技巧。

☑ **紧跟潮流，着眼未来**。本书采用使用广泛的数据库版本——SQL Server 2014 实现，使读者能够紧跟技术发展的脚步。

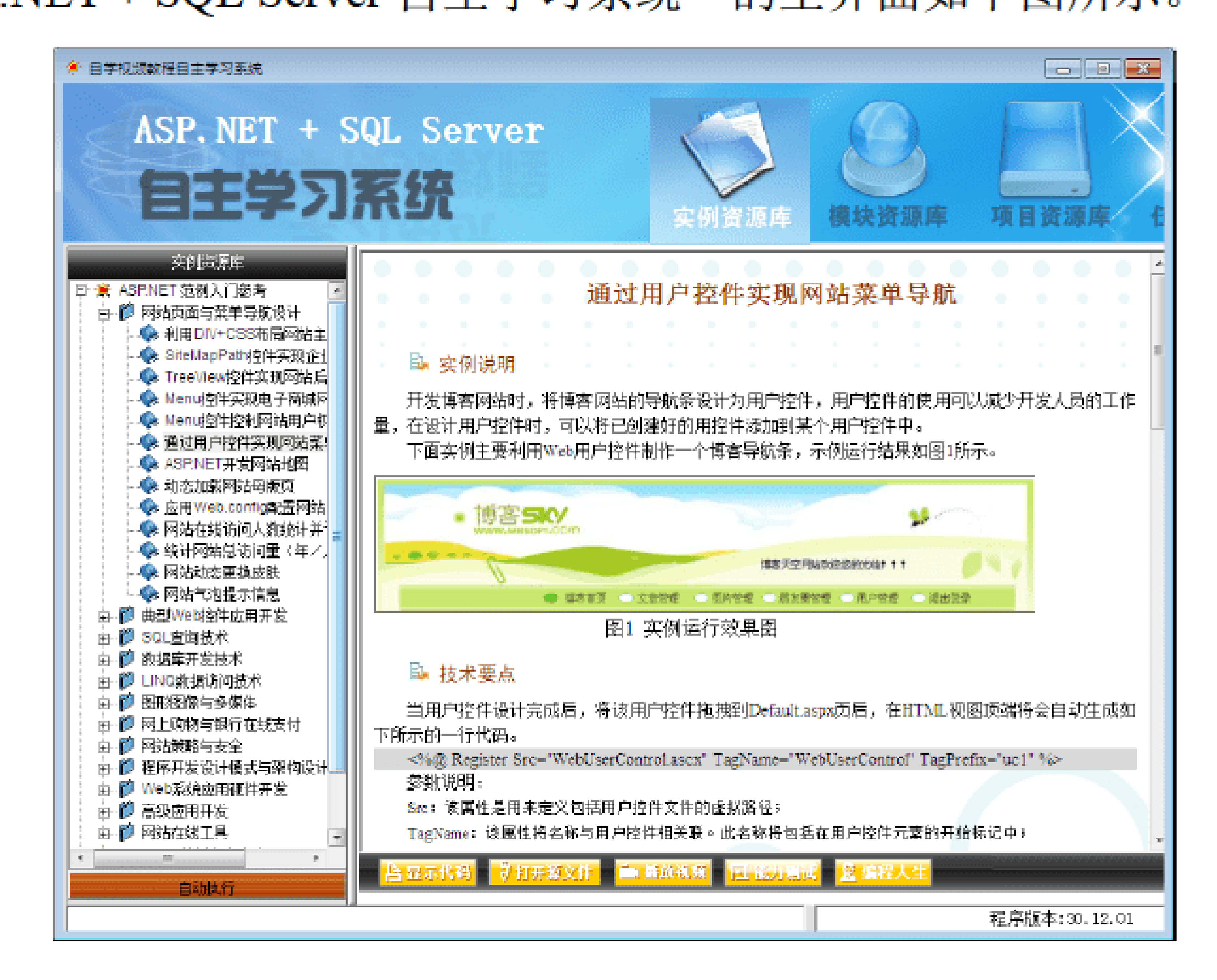
本书资源

为帮助读者学习，本书配备了长达 8 小时（共 71 集）的微课视频讲解。除此以外，还为读者提供了“ASP.NET + SQL Server 自主学习系统”，可以帮助读者快速提升编程水平和解决实际问题的能力。

本书和“ASP.NET + SQL Server 自主学习系统”配合学习流程如图所示。



“ASP.NET + SQL Server 自主学习系统”的主界面如下图所示。





在学习本书的过程中，可以选择实例资源库和项目资源库的相应内容，全面提升个人综合编程技能和解决实际开发问题的能力，为成为软件开发工程师打下坚实基础。

对于数学及逻辑思维能力和英语基础较为薄弱的读者，或者想了解个人数学及逻辑思维能力和编程英语基础的用户，本书提供了数学及逻辑思维能力测试和编程英语能力测试供练习和测试。

读者对象

- ☑ 初学编程的自学者
- ☑ 编程爱好者
- ☑ 大中专院校的老师和学生
- ☑ 相关培训机构的老师和学员
- ☑ 做毕业设计的学生
- ☑ 初、中级程序开发人员
- ☑ 程序测试及维护人员
- ☑ 参加实习的“菜鸟”程序员

读者服务

学习本书时，请先扫描封底的权限二维码（需要刮开涂层）获取学习权限，然后即可免费学习书中的所有线上线下资源。本书所附赠的各类学习资源，读者可登录清华大学出版社网站（www.tup.com.cn），在对应图书页面下获取其下载方式。也可扫描图书封底的“文泉云盘”二维码，获取其下载方式。

致读者

本书由明日科技软件开发团队组织编写。明日科技是一家专业从事软件开发、教育培训以及软件开发教育资源整合的高科技公司，其编写的教材非常注重选取软件开发中的必需、常用内容，同时也很注重内容的易学、方便性以及相关知识的拓展性，深受读者喜爱。其教材多次荣获“全行业优秀畅销品种”“全国高校出版社优秀畅销书”等奖项，多个品种长期位居同类图书销售排行榜的前列。

在编写本书的过程中，我们始终本着科学、严谨的态度，力求精益求精，但错误、疏漏之处在所难免，敬请广大读者批评指正。

感谢您购买本书，希望本书能成为您编程路上的领航者。

“零门槛”编程，一切皆有可能。

祝读书快乐！

编　者

2020 年 8 月



目　录

Contents

第 20 章　腾宇超市管理系统

进入 21 世纪，随着经济的高速发展，各行各业的竞争进入了前所未有的激烈状态，竞争已不再是规模的竞争，还包括技术的竞争、管理的竞争、人才的竞争。超市的竞争也随之进入了一个全新的阶段。仓储店、便利店、特许加盟店、专卖店等都对超市产生了很大的冲击，为了提高物资管理的水平和工作效率，尽可能避免商品流通中各环节出现的问题，为超市开发一套管理系统是十分必要的。本章介绍的超市管理系统主要包括基本档案管理、采购订货管理、仓库入货管理、仓库出货管理、人员管理和部门管理等功能。

通过本章的学习，可以掌握以下要点：

☑　超市管理系统的软件结构和业务流程
☑　超市管理系统的数据库设计
☑　Java 程序连接数据库的方法
☑　设计项目的基本流程

20.1　项目设计思路

20.1.1　功能阐述

超市管理系统是一款辅助超市管理员管理超市的实用性项目，根据超市的日常管理需要，超市管理系统应包括基本档案管理、采购订货管理、仓库入库管理、仓库出库管理、人员管理、部门管理 6 大功能。其中基本档案管理又分为供货商管理、销售商管理、货品档案管理、仓库管理，为管理员提供日常基本信息的功能；采购订货管理模块，用来对日常的采购订货信息进行管理；仓库入库管理，用来管理各种商品入库的信息；仓库出库管理，用来管理商品出库记录；人员管理，用来实现对超市内员工的管理；部门管理，用来实现对超市的各个独立部门进行管理。

20.1.2　系统预览

超市管理系统由多个窗体组成，其中包括系统不可缺少的登录窗体、系统的主窗体、功能模块的子窗体等。下面列出几个典型窗体，其他窗体请参见光盘中的源程序。

超市管理系统的登录窗体如图 20.1 所示。

当用户输入合法的用户名和密码后，单击“登录”按钮，即可进入系统的主窗体，运行结果如图 20.2 所示。



图 20.1　超市管理系统的登录窗体



图 20.2　超市管理系统的主窗体

本程序的主窗体中提供了进入各功能模块的按钮，通过单击这些按钮，可进入各子模块中。各个子功能模块还提供了查询、修改和添加相关信息的操作，例如，修改仓库入库窗体运行结果如图 20.3 所示。

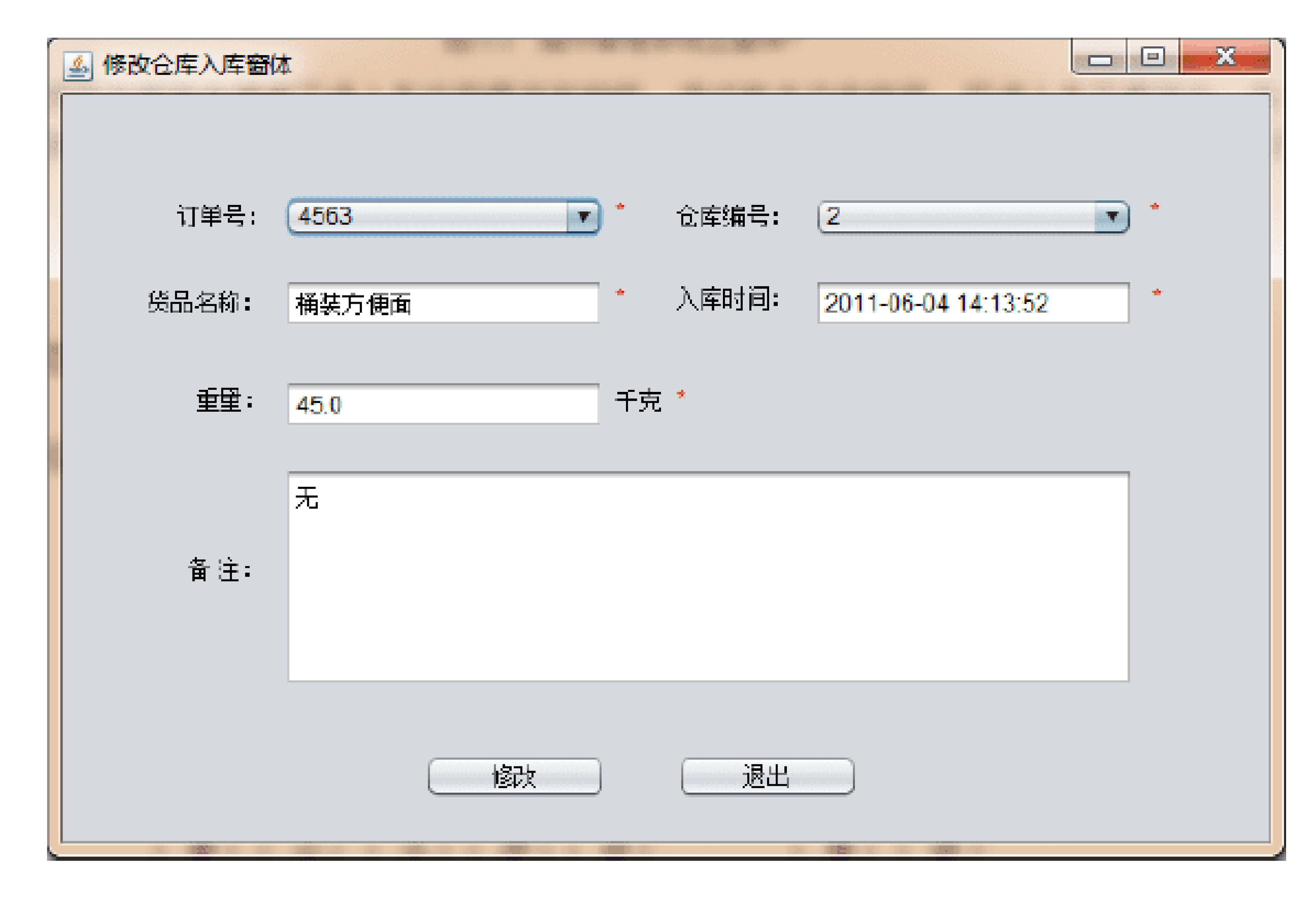


图 20.3　修改仓库入库信息



20.1.3　功能结构

超市管理系统是辅助超市管理员实现对超市的日常管理而设计的，本系统的功能结构如图 20.4 所示。

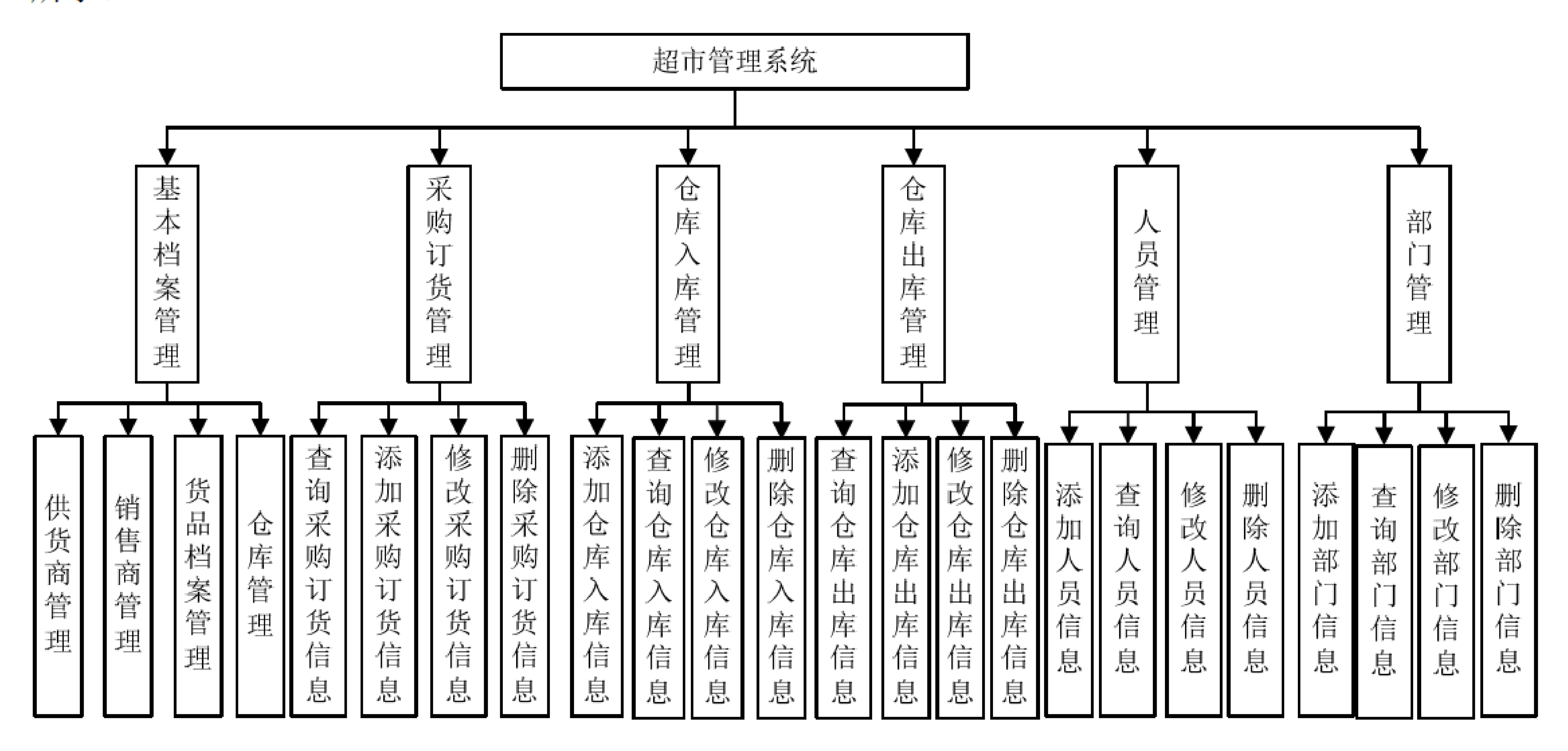


图 20.4　系统功能结构图

20.1.4　文件组织结构

超市管理系统中使用的根目录文件夹是 16，其中包括的文件架构如图 20.5 所示。



图 20.5　超市管理系统的文件架构图

20.2 数据库设计

20.2.1 数据库设计

超市管理系统采用的是 SQL Server 2014 数据库，数据库命名为 db_supermarket，包括的数据表有 tb_basicMessage、tb_contact、tb_depot 等，各数据表描述如图 20.6 所示。

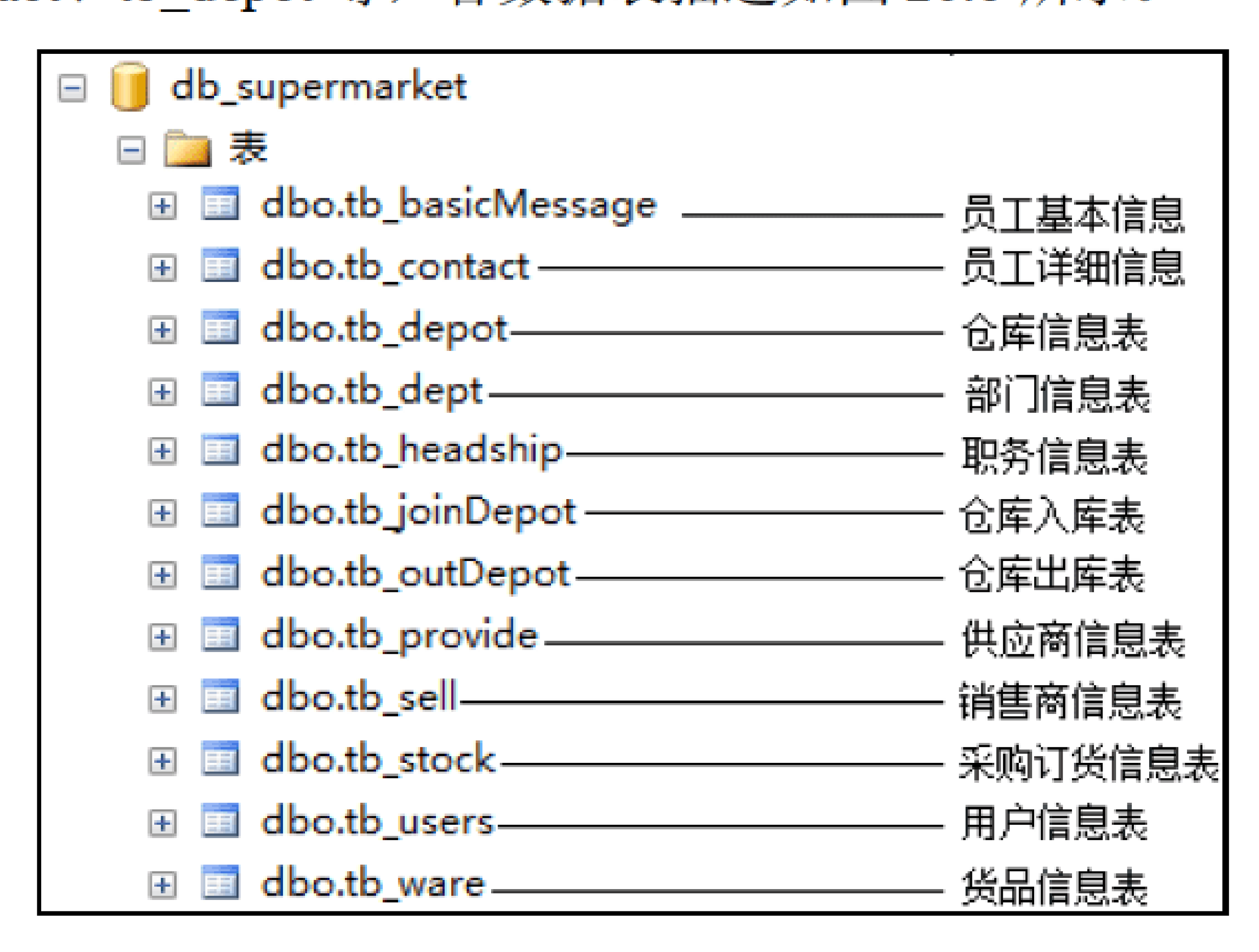


图 20.6 数据库结构

20.2.2 数据表设计

数据表设计是一个非常关键的环节，下面对系统中的数据表结构进行分析。由于篇幅有限，本章只给出了主要的数据表结构。其他数据表结构可参考资源包中的源程序。

1．员工基本信息表（tb_basicMessage）

员工基本信息表包括了员工姓名、年龄、性别、员工所在部门等信息，数据表字段设计如表 20.1 所示。

表 20.1 员工基本信息表设计（tb_basicMessage）

字　段	类　型	额　外	说　明
id	int	自动编号	主键
name	varchar(10)		员工姓名
age	int		员工年龄
sex	varchar(50)		员工性别
dept	int		员工部门，与部门表主键对应
headship	int		员工职务，与职务表主键对应



2．员工详细信息表（tb_contact）

员工详细信息表中保存了员工联系电话、办公电话、传真、邮箱地址、家庭地址等详细信息，数据表字段设计如表 20.2 所示。

表 20.2　员工详细信息表设计（tb_contact）

字　段	类　型	额　外	说　明
id	int	自动编号	主键
hid	int	外键	与员工基本信息表主键对应
contact	varchar(20)		联系电话
officePhone	varchar(30)		办公电话
fax	varchar(20)		传真
email	varchar(50)		邮箱地址
faddress	varchar(50)		家庭地址

3．仓库入库表（tb_ joinDepot）

仓库入库表保存仓库入库信息，其中包括订单编号、仓库编号、货品名称等，数据表字段设计如表 20.3 所示。

表 20.3　仓库入库表设计（tb_ joinDepot）

字　段	类　型	额　外	说　明
id	int	自动编号	主键
oid	vrchar(50)		订货编号
dId	int		仓库编号
wareName	varchar(40)		货品名称
joinTime	varchar(50)		入库时间
weight	float		货品重量
remark	varchar(200)		备注信息

4．用户信息表（tb_users）

用户信息表主要用于存储登录系统用户的用户名与密码信息，数据表字段设计如表 20.4 所示。

表 20.4　用户信息表设计（tb_users）

字　段	类　型	额　外	说　明
id	int	自动编号	主键
userName	varchar(20)		登录系统用户名
passWord	varchar(20)		登录系统密码

5．供应商信息表（tb_provide）

供应商信息表用于保存供应商相关信息，数据表字段设计如表 20.5 所示。

表 20.5　供应商信息表设计（tb_provide）

字　段	类　型	额　外	说　明
id	int	自动编号	主键

续表

字　段	类　型	额　外	说　明
cName	varchar(20)		供应商姓名
address	varchar(40)		供应商地址
linkman	varchar(50)		联系人
linkPhone	varchar(20)		联系电话
faxes	varchar(20)		传真
postNum	varchar(10)		邮箱地址
bankNum	varchar(30)		银行账号
netAddress	varchar(30)		主页
emaillAddress	varchar(50)		邮箱地址
remark	varchar(200)		备注信息

20.3　公共类设计

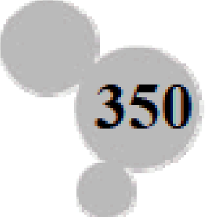
20.3.1　连接数据库

任何系统的设计都离不开数据库，每一步数据库操作都需要与数据库建立连接，为了增加代码的重用性，可以将连接数据库的相关代码保存在一个类中，以便随时调用。创建类 GetConnection，在该类的构造方法中加载数据库驱动，具体代码如下：

```
private Connection con;                                          //定义数据库连接类对象
private PreparedStatement pstm;
private String user="sa";                                        //连接数据库用户名
private String password="";                                      //连接数据库密码
private String className="com.microsoft.sqlserver.jdbc.SQLServerDriver";
//数据库驱动
private String url="jdbc:sqlserver://localhost:1433;DatabaseName=db_supermarket";
//连接数据库的 URL
public GetConnection(){
	try{
		Class.forName(className);
	}catch(ClassNotFoundException e){
		System.out.println("加载数据库驱动失败！");
		e.printStackTrace();
	}
}
```

在该类中定义获取数据库连接方法 getCon()，该方法返回值为 Connection 对象，具体代码如下：

```
public Connection getCon(){
	try {
```



```
			con=DriverManager.getConnection(url,user,password);			//获取数据库连接
		} catch (SQLException e) {
			System.out.println("创建数据库连接失败！");
			con=null;
			e.printStackTrace();
		}
		return con;										//返回数据库连接对象
	}
```

20.3.2　获取当前系统时间类

本系统中多处使用到了应用系统时间的模块，因此可以将获取当前系统时间类作为公共类设计。创建类 GetDate，在该类中定义获取时间方法 getDateTime()，具体代码如下：

```
public static String getDateTime(){								//该方法返回值为 String 类型
	SimpleDateFormat format;
	//SimpleDateFormat 类可以选择任何用户定义的日期-时间格式的模式
	Date date = null;
	Calendar myDate = Calendar.getInstance();
	//Calendar 的方法 getInstance()，以获得此类型的一个通用的对象
	myDate.setTime(new java.util.Date());
	//使用给定的 Date 设置此 Calendar 的时间
	date = myDate.getTime();
	//返回一个表示此 Calendar 时间值（从历元至现在的毫秒偏移量）的 Date 对象
	format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
	//编写格式化时间为“年-月-日  时:分:秒”
	String strRtn = format.format(date);
	//将给定的 Date 格式化为日期/时间字符串，并将结果赋值给给定的 String
	return strRtn;										//返回保存返回值变量
}
```

20.4　登录模块设计

20.4.1　登录模块概述

运行程序，首先进入系统的登录窗体。为了使窗体中的各个组件摆放得更加美观，笔者采用了绝对布局方式，并在窗体中添加了时钟面板来显示时间。运行结果请读者参照 20.1.2 小节中的图 20.1。

20.4.2 实现带背景的窗体

在创建窗体时，需要向窗体中添加面板，之后在面板中添加各种组件。Swing 中代表面板组件的类为 JPanel，该类是以灰色为背景，并且没有任何图片，这样就不能达到很好的美观效果。要实现在窗体中添加背景，就要通过重写 JPanel 面板来实现。

本项目中通过自定义 JPanel 组件来实现，并重写了面板绘制方法，面板绘制方法的声明如下：

```
protected void paintComponent(Graphics graphics)
```

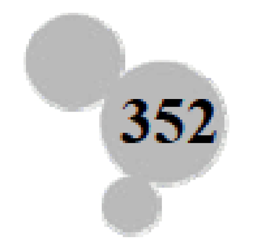
其中，参数 graphics 是指控件中的绘图对象。

例如，本系统中创建的自定义面板 BackgroundPanel，该类继承 JPanel 类，在该类中定义表示背景图片的 Image 对象，重写 paintComponent 方法，实现绘制背景，具体代码如下：

```
public class BackgroundPanel extends JPanel {
    private Image image;                                    //背景图片
    public BackgroundPanel() {
        setOpaque(false);
        setLayout(null);                                    //使用绝对定位布局控件
    }
    /**
     * 设置背景图片对象的方法
     *
     * @param image
     */
    public void setImage(Image image) {
        this.image = image;
    }
    /**
     * 画出背景
     */
    protected void paintComponent(Graphics g) {
        if (image != null) {                                //如果图片已经初始化
            g.drawImage(image, 0, 0, this);                 //画出图片
        }
        super.paintComponent(g);
    }
}
```

20.4.3 登录模块实现过程

登录窗体设计十分简单，由一个“用户名”文本框和一个“密码”文本框组成，为了窗体的美观，笔者还添加了一个显示时钟的面板，该窗体设计如图 20.7 所示。



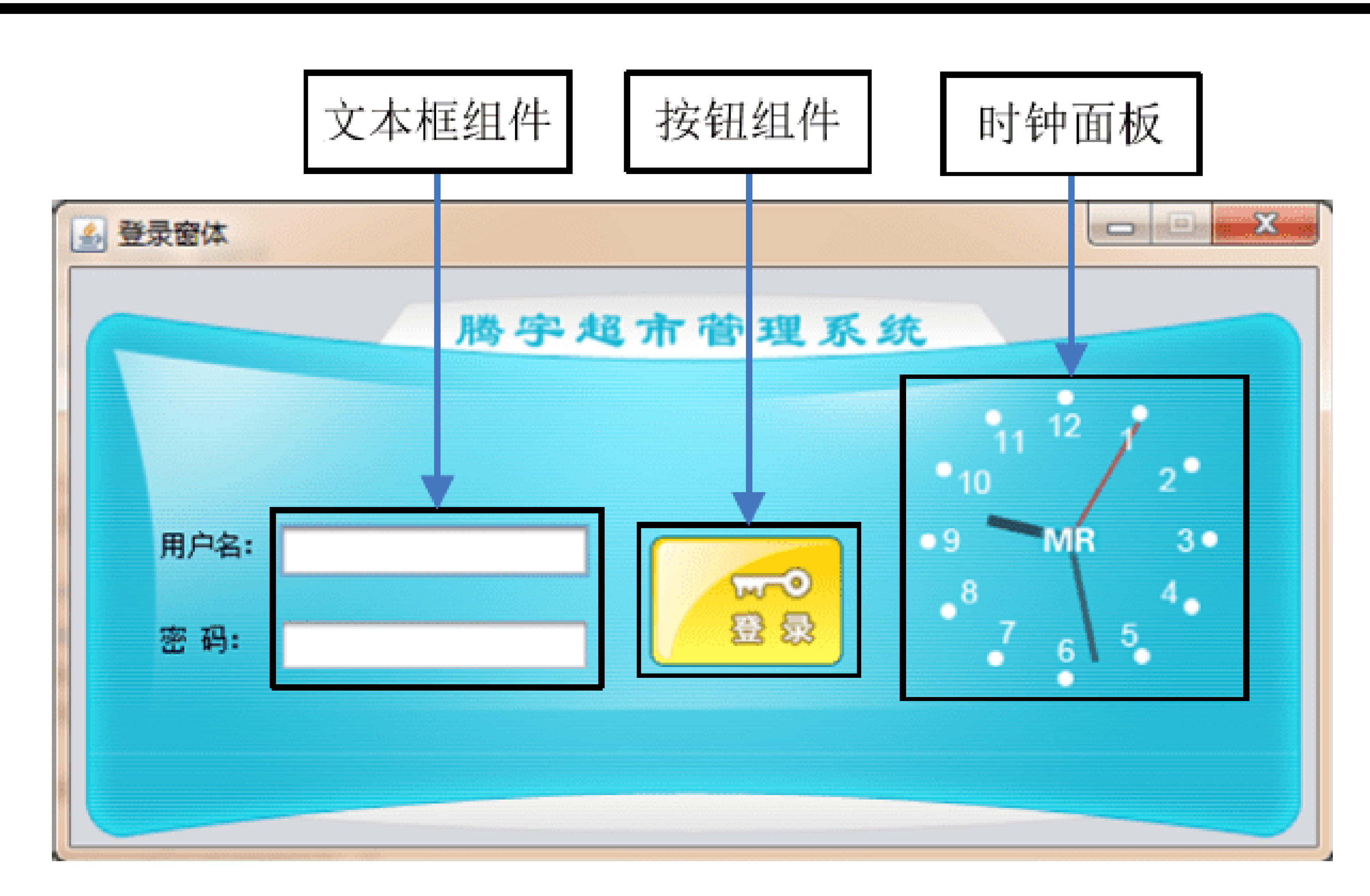


图 20.7　登录窗体设计效果

下面为大家详细地介绍登录模块的实现过程。

（1）实现用户登录操作的数据表是 tb_users，首先创建与数据表对应的 JavaBean 类 User，该类中属性与数据表中字段一一对应，并包含了属性的 setXXX()与 getXXX()方法，具体代码如下：

```
public class User {
	private int id;	//定义映射主键的属性
	private String userName;	//定义映射用户名的属性
	private String passWord;	//定义映射密码的属性
	public int getId() {	//id 属性的 getXXX()方法
		return id;
	}
	public void setId(int id) {	//id 属性的 setXXX()方法
		this.id = id;
	}
	public String getUserName() {
		return userName;
	}
	public void setUserName(String userName) {
		this.userName = userName;
	}
	public String getPassWord() {
		return passWord;
	}
	public void setPassWord(String passWord) {
		this.passWord = passWord;
	}
}
```

（2）由于本系统的主窗体中显示了当前登录系统的用户名，而当前登录的用户对象是在登录窗体中查询出来的，为了实现两个窗体间的通信，可以创建保存用户会话的 Session 类，该类中包含有 User 对象的属性，并含有该属性的 setXXX()与 getXXX()方法，代码如下：

```
public class Session {
	private static User user;	//User 对象属性
```

```
        public static User getUser() {                    //属性的 getXXX()方法
                return user;
        }
        public static void setUser(User user) {           //属性的 setXXX()方法
                 Session.user = user;
        }
}
```

（3）定义类 UserDao，在该类中实现按用户名与密码查询用户方法 getUser()，该方法的返回值为 User 对象，具体代码如下：

```
GetConnection connection = new GetConnection();
Connection conn = null;
//编写按用户名和密码查询用户的方法
public User getUser(String userName,String passWord){
        User user = new User();                           //创建 JavaBean 对象
        conn = connection.getCon();                       //获取数据库连接
        try {
                String sql = "select * from tb_users where userName = ? and passWord = ?";
                //定义查询预处理语句
                PreparedStatement statement = conn.prepareStatement(sql);
                //实例化 PreparedStatement 对象
                statement.setString(1, userName);         //设置预处理语句参数
                statement.setString(2, passWord);
                ResultSet rest = statement.executeQuery(); //执行预处理语句
                while(rest.next()){
                        user.setId(rest.getInt(1));       //应用查询结果设置对象属性
                        user.setUserName(rest.getString(2));
                        user.setPassWord(rest.getString(3));
                }
        } catch (SQLException e) {
                e.printStackTrace();
        }
        return user;                                      //返回查询结果
}
```

（4）在“登录”按钮的单击事件中，调用判断用户是否合法方法 getUser()，实现如果用户输入的用户名与密码合法将转发至系统主窗体，如果用户输入了错误的用户名与密码，则给出相应的提示，具体代码如下：

```
enterButton.addActionListener(new ActionListener() {              //按钮的单击事件
        public void actionPerformed(ActionEvent e) {
                UserDao userDao = new UserDao();                  //创建保存有操作数据库类对象
                //以用户添加的用户名与密码为参数调用查询用户方法
                User user
                = userDao.getUser(userNameTextField.getText(),passwordField.getText());
                if(user.getId()>0){                               //判断用户编号是否大于 0
                        Session.setUser(user);                    //设置 Session 对象的 User 属性值
                        RemoveButtomFrame frame = new RemoveButtomFrame();      //创建主窗体对象
                        frame.setVisible(true);                   //显示主窗体
```

```
                Enter.this.dispose();                                       //销毁登录窗体
            }
            else{                                                           //如果用户输入的用户名与密码错误
                JOptionPane.showMessageDialog(getContentPane(), "用户名或密码错误");    //给出提示信息
                userNameTextField.setText("");                              // “用户名”文本框设置为空
                passwordField.setText("");                                  // “密码”文本框设置为空
            }
        }
});
```

20.5　主窗体设计

20.5.1　主窗体概述

成功登录系统后，即可进入系统的主窗体。系统的主窗体中以移动面板的形式显示了各功能按钮，并在初始化状态中显示了基本档案管理模块的相关功能，并为用户提供了时钟和日历面板。主窗体运行结果如图 20.8 所示。



图 20.8　主窗体运行结果

20.5.2　平移面板控件

在主窗体中笔者添加了移动面板控件，移动面板在水平方向添加了多个控件，通过左右平移两个按钮可以调整显示内容。在窗体中添加平移面板不仅可以增加窗体的灵活性，还能够提升窗体的美观效果。实现平移面板关键在于控制滚动面板中滚动条的当前值，就需要获取滚动面板的滚动条与设置滚动条当前值的相关知识，下面分别进行介绍。

☑ 获取滚动面板的水平滚动条

滚动面板包含水平和垂直两个方向的滚动条，通过适当的方法可以获取它们，下面的方法可以获取控制视口的水平视图位置的水平滚动条。方法声明如下：

```
public JScrollBar getHorizontalScrollBar()
```

☑ 获取滚动条当前值

滚动条的控制对象就是当前值，这个值控制着滚动条滑块的位置和滚动面板视图的位置。可以通过 getValue()方法来获取这个值，方法声明如下：

```
public int getValue()
```

☑ 设置滚动条当前值

```
public void setValue(int value)
```

其中，参数 value 指滚动条新的当前值。

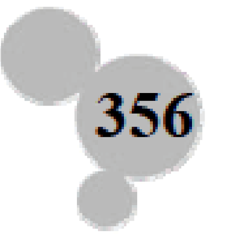
创建成功滚动面板后，将按钮添加到滚动面板即可，本系统实现滚动面板的类为 SmallScrollPanel，该类是一个面板类，在该类的构造方法中初始化面板滚动事件处理器，代码如下：

```
public SmallScrollPanel() {
    scrollMouseAdapter = new ScrollMouseAdapter();                 //初始化处理器
    //初始化程序用图
    icon1 = new ImageIcon(getClass().getResource("top01.png"));
    icon2 = new ImageIcon(getClass().getResource("top02.png"));
    setIcon(icon1);                                                //设置用图
    setIconFill(BOTH_FILL);                                        //将图标拉伸适应界面大小
    initialize();                                                  //调用初始化方法
}
```

在 SmallScrollPanel 类的初始化方法中设置面板布局，并在窗体中添加左侧和右侧的微调按钮，具体代码如下：

```
private void initialize() {
    BorderLayout borderLayout = new BorderLayout();
    borderLayout.setHgap(0);
    this.setLayout(borderLayout);                                  //设置布局管理器
    this.setSize(new Dimension(300, 84));
    this.setOpaque(false);                                         //使控件透明
    //添加滚动面板到界面居中位置
    this.add(getAlphaScrollPanel(), BorderLayout.CENTER);
    //添加左侧微调按钮
    this.add(getLeftScrollButton(), BorderLayout.WEST);
    //添加右侧微调按钮
    this.add(getRightScrollButton(), BorderLayout.EAST);
}
```

在平移面板中左右侧的两个箭头形状平移按钮，为两个添加背景的按钮，将该按钮的边框去掉，



就可显示大家看到的效果。下面以左侧微调按钮为例，介绍微调按钮的实现代码：

```
private JButton getLeftScrollButton() {
    if (leftScrollButton == null) {
        leftScrollButton = new JButton();
        //创建按钮图标
        ImageIcon icon1 = new ImageIcon(getClass().getResource(
                "/com/mingrisoft/frame/buttonIcons/zuoyidongoff.png"));
        //创建按钮图标 2
        ImageIcon icon2 = new ImageIcon(getClass().getResource(
                "/com/mingrisoft/frame/buttonIcons/zuoyidongon.png"));
        leftScrollButton.setOpaque(false);                    //按钮透明
        //设置边框
        leftScrollButton.setBorder(createEmptyBorder(0, 10, 0, 0));
        //设置按钮图标
        leftScrollButton.setIcon(icon1);
        leftScrollButton.setPressedIcon(icon2);
        leftScrollButton.setRolloverIcon(icon2);
        //取消按钮内容填充
        leftScrollButton.setContentAreaFilled(false);
        //设置初始大小
        leftScrollButton.setPreferredSize(new Dimension(38, 0));
        //取消按钮焦点功能
        leftScrollButton.setFocusable(false);
        //添加滚动事件监听器
        leftScrollButton.addMouseListener(scrollMouseAdapter);
    }
    return leftScrollButton;
}
```

创建左右微调按钮的事件监听器，实现当用户单击左右微调按钮时，移动面板，具体代码如下：

```
private final class ScrollMouseAdapter extends MouseAdapter implements
        Serializable {
    private static final long serialVersionUID = 5589204752770150732L;
    JScrollBar scrollBar = getAlphaScrollPanel().getHorizontalScrollBar();
    //获取滚动面板的水平滚动条
    private boolean isPressed = true;                    //定义线程控制变量
    public void mousePressed(MouseEvent e) {
        Object source = e.getSource();                   //获取事件源
        isPressed = true;
        if (source == getLeftScrollButton()) {    //判断事件源是左侧按钮还是右侧按钮，并执行相应操作
            scrollMoved(-1);
        } else {
            scrollMoved(1);
        }
    }
    /**
     * 移动滚动条的方法
     * @param orientation
```

```
     *       移动方向 -1 是左或上移动，1 是右或下移动
     */
    private void scrollMoved(final int orientation) {
        new Thread() {                                      //开辟新的线程
            private int oldValue = scrollBar.getValue();    //保存原有滚动条的值

            public void run() {
                while (isPressed) {                         //循环移动面板
                    try {
                        Thread.sleep(10);
                    } catch (InterruptedException e1) {
                        e1.printStackTrace();
                    }
                    oldValue = scrollBar.getValue();        //获取滚动条当前值
                    EventQueue.invokeLater(new Runnable() {
                        public void run() {
                            scrollBar.setValue(oldValue + 3 * orientation);
                            //设置滚动条移动 3 个像素
                        }
                    });
                }
            }
        }.start();
    }
    public void mouseExited(java.awt.event.MouseEvent e) {
        isPressed = false;
    }

    @Override
    public void mouseReleased(MouseEvent e) {
        isPressed = false;
    }
}
```

平移面板 SmallScrollPanel 类的设计效果如图 20.9 所示。

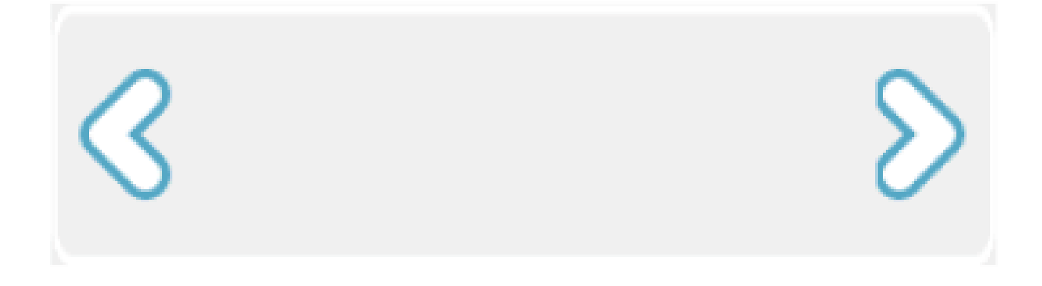
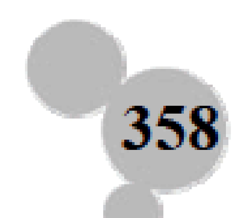


图 20.9　平移面板设计效果

20.5.3　主窗体实现过程

主窗体由多个面板组成，除了前面介绍过的功能按钮面板、时钟面板、日历面板外，还包括功能区面板，与主窗体中的其他面板不同，功能区面板是随时更换的，当用户单击不同的功能按钮，系统通过显示不同的面板来实现窗体内容的随时更换，设计效果如图 20.10 所示。



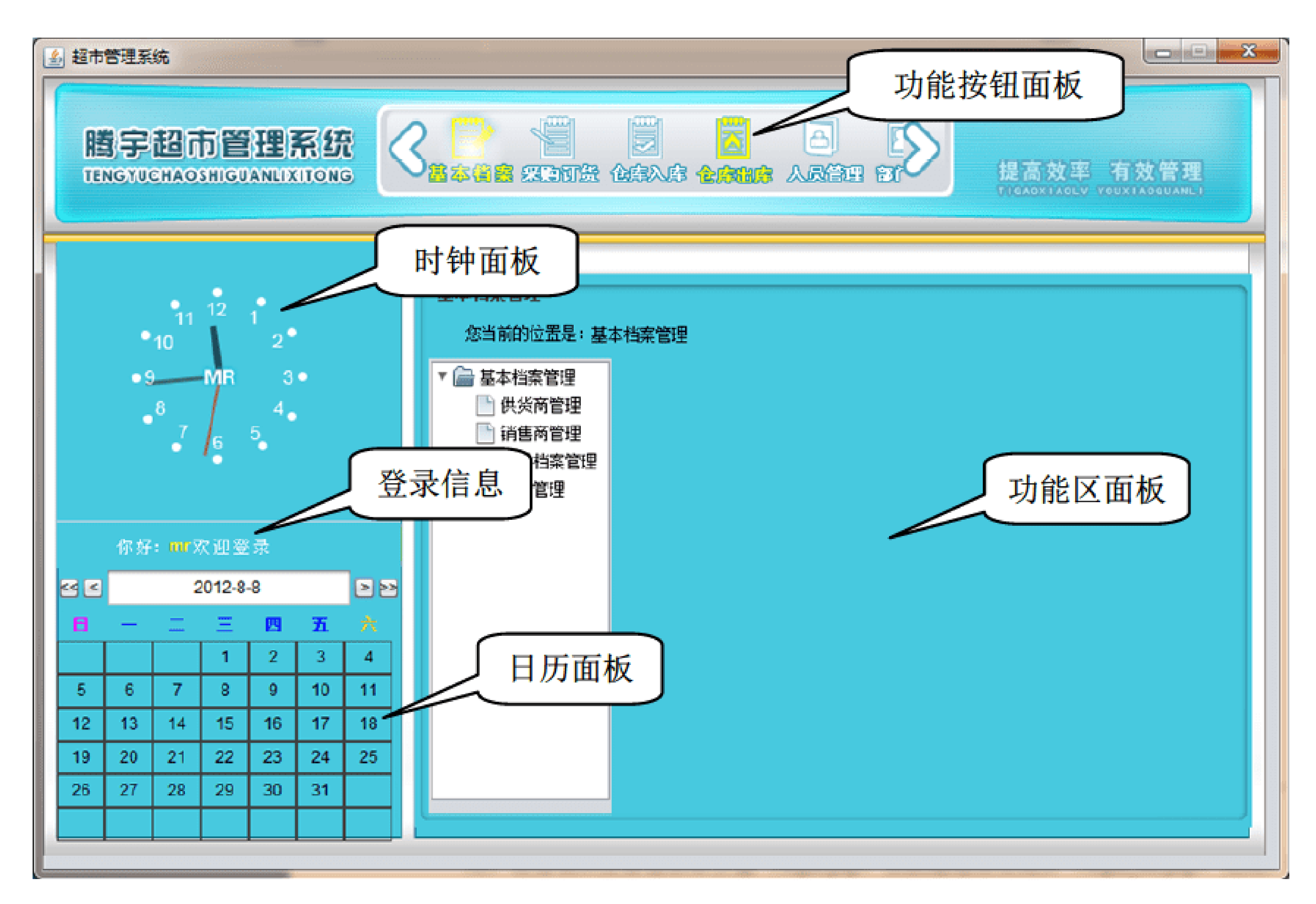


图 20.10　主窗体设计效果

下面介绍在主窗体的实现过程中几个重要的实现过程。

（1）通过如图 20.10 所示的主窗体设计效果可以看到，在主窗体中显示了当前登录的用户名，实现显示当前登录用户名代码如下：

```
User user = Session.getUser();                                  //获取登录用户对象
String info = "<html><body>" + "<font color=#FFFFFF>你 好：</font>"
            + "<font color=yellow><b>" + user.getUserName() + "</b></font>"
            + "<font color=#FFFFFF> 欢 迎 登 录</font>" + "</body></html>";
//定义窗体显示内容
clockpanel.add(getPanel());
JLabel label = new JLabel(info);                                 //定义显示指定内容的标签对象
```

（2）创建完成如图 20.9 所示的平移面板后，需要创建按钮组面板，再将按钮组面板添加到平移面板，才实现了主窗体中显示的效果，按钮组面板采用网格布局，设计效果如图 20.11 所示。



图 20.11　按钮组面板设计效果

按钮组面板实现代码如下：

```
public BGPanel getJPanel() {
    if (jPanel == null) {
        GridLayout gridLayout = new GridLayout();               //定义网格布局管理器
        gridLayout.setRows(1);                                  //设置网格布局管理器的行数
```

```
            gridLayout.setHgap(0);                              //设置组件间水平间距
            gridLayout.setVgap(0);                              //设置组件间垂直间距
            jPanel = new BGPanel();
            jPanel.setLayout(gridLayout);                       //设置布局管理器
            jPanel.setPreferredSize(new Dimension(400, 50));    //设置初始大小
            jPanel.setOpaque(false);
            jPanel.add(getWorkSpaceButton(), null);             //添加按钮
            jPanel.add(getProgressButton(), null);
            jPanel.add(getrukuButton(), null);
            jPanel.add(getchukuButton(), null);
            jPanel.add(getPersonnelManagerButton(), null);
            jPanel.add(getDeptManagerButton(), null);
            if (buttonGroup == null) {
                buttonGroup = new ButtonGroup();
            }
            // 把所有按钮添加到一个组控件中
            buttonGroup.add(getProgressButton());
            buttonGroup.add(getWorkSpaceButton());
            buttonGroup.add(getrukuButton());
            buttonGroup.add(getchukuButton());
            buttonGroup.add(getPersonnelManagerButton());
            buttonGroup.add(getDeptManagerButton());
        }
        return jPanel;
}
```

（3）本系统中将平移面板中的各个按钮都封装在单独的方法中，下面以“基本档案”按钮为例，介绍平移面板中的各按钮的实现代码：

```
private GlassButton getWorkSpaceButton() {
        if (workSpaceButton == null) {
            workSpaceButton = new GlassButton();
            workSpaceButton.setActionCommand("基本档案管理");   //设置按钮的动作命令
            workSpaceButton.setIcon(new ImageIcon(getClass().getResource(
                        "/com/mingrisoft/frame/buttonIcons/myWorkSpace.png")));
            //定义按钮的初始化背景
            ImageIcon icon = new ImageIcon(getClass().getResource(
                        "/com/mingrisoft/frame/buttonIcons/myWorkSpace2.png"));
            //创建图片对象
            workSpaceButton.setRolloverIcon(icon);              //设置按钮的翻转图片
            workSpaceButton.setSelectedIcon(icon);              //设置按钮被选中时显示图片
            workSpaceButton.setSelected(true);
            workSpaceButton.addActionListener(new toolsButtonActionAdapter());
            //按钮的监听器
        }
        return workSpaceButton;
}
```

20.6　采购订货模块设计

20.6.1　采购订货模块概述

在超市的日常管理活动中，对于商品的采购和订货是不可缺少的。当用户单击平移面板中的“采购订货”按钮，即可进入采购订货模块，该模块中以表格的形式显示采购订货信息，在采购订货模块中还包括添加采购订货信息、修改采购订货信息、删除采购订货信息功能，运行效果如图 20.12 所示。



图 20.12　采购订货模块运行效果

20.6.2　在表格中添加按钮

表格用于显示复合数据，其中可以指定表格的表头和表文，默认的表格控件完全是以文本方式显示目标数据，要实现在表格中添加按钮或其他组件就要通过设置自定义的渲染器来实现，表格的渲染器通过 TableCellRenderer 接口实现，该接口中定义了 getTableCellRendererComponent()方法，这个方法将被表格控件回调来渲染指定的单元格控件。重写这个方法并在方法体中控制单元格的渲染，就可以把按钮作为表格的单元格控件。该方法的声明如下：

```
Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected,boolean hasFocus,
int row, int column)
```

方法中的参数说明如表 20.6 所示。

表 20.6　getTableCellRendererComponent()方法的参数说明

字　段	类　型
table	要求渲染器绘制的 JTable；可以为 NULL
value	要呈现的单元格的值。由具体的渲染器解释和绘制该值。例如，如果 value 是字符串 "TRUE"，则它可呈现为字符串，或者也可呈现为已选中的复选框。NULL 是有效值
isSelected	如果使用选中样式的高亮显示来呈现该单元格，则为 TRUE；否则为 FALSE
hasFocus	如果为 TRUE，则适当地呈现单元格。例如，在单元格上放入特殊的边框，如果可以编辑该单元格，则以彩色呈现它，用于表示正在进行编辑
row	要绘制的单元格的行索引。绘制表头时，row 值是-1
column	要绘制的单元格的列索引

例如，本模块中，设置“是否入库”列的渲染器，代码如下：

```
table.getColumn("是否入库").setCellRenderer(new ButtonRenderer());      //设置指定列的渲染器
```

20.6.3　添加采购订货信息实现过程

用户单击采购订货窗体中的“添加”按钮，即可弹出添加采购订货窗体，该窗体运行结果如图 20.13 所示。



图 20.13　添加采购订货窗体运行结果

下面详细地介绍添加采购订货窗体的实现过程。

（1）创建与采购订货信息表 tb_stock 对应的 JavaBean 对象 Stock，该类中的属性与 tb_stock 表中的字段一一对应，并包括了各属性的 setXXX()与 getXXX()方法，具体代码如下：

```
public class Stock {
        private int id;
        private String sName;
        private String orderId;
        private String consignmentDate;
        private String baleName;
        private String count;
        private float money;
        private String lairage;
```



```
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    …//省略了其他属性的 setXXX()与 getXXX()方法
}
```

（2）定义对采购订货信息表 tb_stock 中数据进行操作类 StockDao，其中添加采购订货信息方法 insertStock()，该方法以 Stock 为对象，具体代码如下：

```
public void insertStock(Stock stock) {
    conn = connection.getCon();                              //获取数据库连接
    try {
        PreparedStatement statement = conn
                .prepareStatement("insert into tb_stock values(?,?,?,?,?,?)");
        //定义查询数据的 SQL 语句
        statement.setString(1,stock.getsName());             //设置预处理语句参数
        statement.setString(2,stock.getOrderId());
        statement.setString(3,stock.getConsignmentDate());
        statement.setString(4,stock.getBaleName());
        statement.setString(5,stock.getCount());
        statement.setFloat(6,stock.getMoney());
        statement.executeUpdate();                           //执行插入操作
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

（3）在添加采购订货窗体的“添加”按钮的单击事件中，实现判断用户填写的信息是否合法，再将这些信息保存到数据库中，具体代码如下：

```
JButton insertButton = new JButton("添加");
insertButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        StockDao dao = new StockDao();                       //定义操作数据表方法
        String oId = orderIdTextField.getText();             //获取用户添加的订单号
        String wname = nameTextField.getText();              //获取用户添加的客户名称
        String wDate = dateTextField.getText();              //获取用户添加的交货日期
        String count = countTextField.getText();             //获取用户添加的商品数量
        String bName = wNameTextField.getText();             //获取用户添加的货品名称
        String money = moneyTextField.getText();             //获取用户添加的货品金额
        int countIn = 0;
        float fmoney = 0;
        if((oId.equals(""))||(wname.equals("")) ||(wDate.equals("")) ||
            (count.equals("")) || (money.equals(""))){       //判断用户添加的信息是否完整
        JOptionPane.showMessageDialog(getContentPane(), "请将带星号的内容填写完整！",
                "信息提示框", JOptionPane.INFORMATION_MESSAGE);
        //给出提示信息
```

```
                return;                                         //退出程序
            }
            try{
                countIn = Integer.parseInt(count);              //将用户添加的数量转换为整型
                fmoney = Float.parseFloat(money);
            }catch (Exception ee) {
                JOptionPane.showMessageDialog(getContentPane(), "要输入数字！",
                        "信息提示框", JOptionPane.INFORMATION_MESSAGE);
                return;
            }
            Stock stock = new Stock();                          //定义与数据表对应的 JavaBean 对象
            stock.setsName(wname);                              //设置对象属性
            stock.setBaleName(bName);
            stock.setConsignmentDate(wDate);
            stock.setCount(count);
            stock.setMoney(fmoney);
            stock.setOrderId(oId);
            dao.insertStock(stock);
            //调用数据库添加方法
            JOptionPane.showMessageDialog(getContentPane(), "数据添加成功！",
                    "信息提示框", JOptionPane.INFORMATION_MESSAGE);       //提示信息
        }
});
```

20.6.4 搜索采购订货信息实现过程

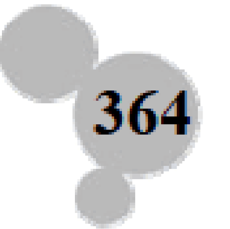
在采购订货模块中，添加了按指定条件搜索采购订货信息功能，用户可按照自己的需求指定查询条件。搜索采购订货窗体运行结果如图 20.14 所示。



图 20.14 搜索采购订货窗体

下面介绍搜索采购订货信息的具体实现过程。

（1）在搜索采购订货窗体中，为用户提供按“货品名称”“订单号”“交货时间”搜索指定采购订



货信息。下面以按货品名称查询采购订货信息为例，为大家介绍查询数据库方法，具体代码如下：

```
public List selectStockBySName(String sName) {
	List list = new ArrayList<Stock>();				//定义保存查询结果的 List 对象
	conn = connection.getCon();					//获取数据库连接
	int id = 0;
	try {
		Statement statement = conn.createStatement();		//实例化 Statement 对象
		//定义查询语句，获取查询结果集
		ResultSet rest = statement.executeQuery("select * from tb_stock where sName ='"+sName+"'");
		while (rest.next()) {					//循环遍历查询结果集
			Stock stock = new Stock();			//定义与数据表对象的 JavaBean 对象
			stock.setId(rest.getInt(1));			//应用查询结果设置 JavaBean 属性
			stock.setsName(rest.getString(2));
			stock.setOrderId(rest.getString(3));
			stock.setConsignmentDate(rest.getString(4));
			stock.setBaleName(rest.getString(5));
			stock.setCount(rest.getString(6));
			stock.setMoney(rest.getFloat(7));
			list.add(stock);				//将 JavaBean 对象添加到集合
		}
	} catch (SQLException e) {
		e.printStackTrace();
	}
	return list;							//返回查询集合
}
```

（2）当用户单击“搜索”按钮时，首先将表格中的数据全部删除，再将满足条件的数据填写到表格中，关键代码如下：

```
JButton findButton = new JButton("搜索");
findButton.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		dm.setRowCount(0);					//将表格内容清空
		String condition = comboBox.getSelectedItem().toString();
		//获取用户选择的查询条件
		String conditionText = conditionTextField.getText();	//获取用户添加的查询条件
		if(conditionText.equals("")){				//如果用户没有添加查询条件
			JOptionPane.showMessageDialog(getParent(), "请输入查询条件！",
					"信息提示框", JOptionPane.INFORMATION_MESSAGE); //给出提示信息
			return;						//退出程序
		}
		if(condition.equals("货品名称")){			//如果用户选择按货品名称进行搜索
		    List list = dao.selectStockBySName(conditionText);
		    //调用按货品名称查询数据方法
		    for(int i= 0;i<list.size();i++){			//循环遍历查询结果
			    Stock stock = (Stock)list.get(i);
			     String oid = stock.getOrderId();		//获取订单号信息
			int id = dao.selectJoinStockByOid(oid);		//根据订单号查询入库信息
			if(id <=0){					//如果该订单的货品在入库表中不存在
```

```
                dm.addRow(new Object[]{"入库",stock.getId(),stock.getsName(),stock.getOrderId(),
                        stock. getConsignmentDate(),stock.getBaleName(),
                        stock.getMoney(),stock.getCount()});   //向表格中添加数据
            }
            else{                                        //如果指定订单号的货品名称在入库表中存在
                dm.addRow(new Object[]{"已经入库",stock.getId(),stock.getsName(),stock.getOrderId(),
                        stock.getConsignmentDate(),stock.getBaleName(),
                        stock.getMoney(),stock.getCount()});
            }
        }
    }
```

20.6.5　修改采购订货信息实现过程

采购订货模块中提供了修改采购订货信息功能，当用户在显示采购订货信息的表格中选择要修改的信息后，单击窗体中的“修改”按钮，即可打开修改采购订单窗体，运行结果如图 20.15 所示。



图 20.15　修改采购订单窗体

下面详细地介绍修改采购订单窗体的实现过程。

（1）创建修改采购订货信息方法 updateStock()，该方法以 Stock 对象作为参数，具体代码如下：

```
public void updateStock(Stock stock) {
    conn = connection.getCon();                              //获取数据库连接
    try {
        String sql = "update tb_stock set sName=?,orderId=?,consignmentDate=?," +
                "baleName=?,count=?,money=? where id =?";   //定义修改数据表方法
        PreparedStatement statement = conn.prepareStatement(sql);
                                                             //获取 PreparedStatement 对象
        statement.setString(1, stock.getsName());            //设置预处理语句参数值
        statement.setString(2, stock.getOrderId());
        statement.setString(3, stock.getConsignmentDate());
        statement.setString(4, stock.getBaleName());
        statement.setString(5, stock.getCount());
        statement.setFloat(6, stock.getMoney());
        statement.setInt(7, stock.getId());
        statement.executeUpdate();                           //执行更新语句
    } catch (SQLException e) {
```

```
            e.printStackTrace();
        }
}
```

（2）要实现修改采购订货信息，首先将要修改的内容查询出来，并显示在窗体中。这样才能实现修改操作，首先编写按编号查询采购订货信息方法 selectStockByid()，具体代码如下：

```
public Stock selectStockByid(int id) {
        Stock stock = new Stock();                                      //定义与数据库对应的 JavaBean 对象
        conn = connection.getCon();                                     //获取数据库连接
        try {
                Statement statement = conn.createStatement();
                String sql = "select * from tb_stock where id = " + id;  //定义查询 SQL 语句
                ResultSet rest = statement.executeQuery(sql);            //执行查询语句获取查询结果集
                while (rest.next()) {                                    //循环遍历查询结果集
                        stock.setId(id);                                 //应用查询结果设置对象属性
                        stock.setsName(rest.getString(2));
                        stock.setOrderId(rest.getString(3));
                        stock.setConsignmentDate(rest.getString(4));
                        stock.setBaleName(rest.getString(5));
                        stock.setCount(rest.getString(6));
                        stock.setMoney(rest.getFloat(7));
                }
        } catch (SQLException e) {
                e.printStackTrace();
        }
        return stock;                                                   //返回 Stock 对象
}
```

（3）由于显示采购订单窗体与修改采购订单窗体是两个独立的窗体，用户需要在显示采购订单窗体中选择要修改的信息，系统会将指定采购订货信息的编号写入文本文件中，之后在修改采购订单窗体中读取出来，这样就可实现在修改采购订单窗体中显示要修改的订货信息。在显示采购订单窗体中，将用户选择的采购订货信息保存在文本文件中，具体代码如下：

```
JButton updateButton = new JButton("修改");
updateButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                int row = table.getSelectedRow();                        //获取用户选中表格的行数
                        if (row < 0) {
                                JOptionPane.showMessageDialog(getParent(), "没有选择要修改的数据！",
                                                "信息提示框", JOptionPane.INFORMATION_MESSAGE);
                                return;
                        } else {
                                File file = new File("filedd.txt");      //创建文件对象
                                try {
                                        String column = dm.getValueAt(row, 1).toString();
                                                                         //获取表格中的数据
                                        file.createNewFile();            //新建文件
                                        FileOutputStream out = new FileOutputStream(file);
```

```
                    out.write((Integer.parseInt(column)));          //将数据写入文件中
                    UpdateStockFrame frame = new UpdateStockFrame();
                                                                    //创建修改信息窗体
                    frame.setVisible(true);
                    out.close();                                    //将流关闭
                    repaint();
                } catch (Exception ee) {
                    ee.printStackTrace();
                }
            }
        }
});
```

（4）在修改采购订单窗体 UpdateStockFrame 中，读取用户写在文本文件中保存的要修改的采购订货信息的编号，再按照这个编号查询要修改的采购订货信息对象，将该对象的信息显示在窗体中，关键代码如下：

```
try {
    File file = new File("filedd.txt");                             //创建文件对象
    FileInputStream fin = new FileInputStream(file);                //创建文件输入流对象
    int count =   fin.read();                                       //读取文件中数据
    stock = dao.selectStockByid(count);                             //调用按编号查询数据方法
    file.delete();                                                  //删除文件
 } catch (Exception e) {
        e.printStackTrace();
}
JLabel orderIdLabel = new JLabel("订单号：");
orderIdLabel.setBounds(59, 55, 60, 15);
contentPane.add(orderIdLabel);
orderIdTextField = new JTextField();                                //创建文本框对象
orderIdTextField.setText(stock.getOrderId());                       //设置文本框对象内容
orderIdTextField.setBounds(114, 50, 164, 25);
contentPane.add(orderIdTextField);                                  //将文本框对象添加到面板中
orderIdTextField.setColumns(10);
...//省略了设置窗体其他内容的代码
```

（5）在修改采购订单窗体的“修改”按钮中，调用修改采购订货信息方法，将用户修改的信息保存到数据库中，具体代码如下：

```
JButton insertButton = new JButton("修改");
insertButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        StockDao dao = new StockDao();                              //创建保存有修改方法的类对象
        String old = orderIdTextField.getText();                    //获取用户填写订单数据
        String wname = nameTextField.getText();                     //获取用户填写的客户名信息
        String wDate = dateTextField.getText();                     //获取用户填写的交货日期信息
        String count = countTextField.getText();
        String bName = wNameTextField.getText();
        String money = moneyTextField.getText();
```

```
            int countIn = 0;
            float fmoney = 0;
            if((oId.equals(""))||(wname.equals("")) ||(wDate.equals("")) ||
                    (count.equals("")) || (money.equals(""))){              //判断用户是否将信息添加完整
                    JOptionPane.showMessageDialog(getContentPane(), "请将带星号的内容填写完整！",
                                    "信息提示框", JOptionPane.INFORMATION_MESSAGE);    //给出提示信息
                    return;
            }
            try{
                    countIn = Integer.parseInt(count);                  //将用户填写的数量转换为整数
                    fmoney = Float.parseFloat(money);
            }catch (Exception ee) {
                    JOptionPane.showMessageDialog(getContentPane(), "要输入数字！",
                                    "信息提示框", JOptionPane.INFORMATION_MESSAGE);
                    //如果有异常抛出给出提示信息
                    return;
            }
            stock.setsName(wname);                                    //将设置采购订货信息属性
            stock.setBaleName(bName);
            stock.setConsignmentDate(wDate);
            stock.setCount(count);
            stock.setMoney(fmoney);
            stock.setOrderId(oId);
            dao.updateStock(stock);                                   //调用修改信息方法
            JOptionPane.showMessageDialog(getContentPane(), "数据添加成功！",
                            "信息提示框", JOptionPane.INFORMATION_MESSAGE);
        }
});
```

20.6.6　删除采购订货信息实现过程

如果要删除某采购订货信息，可以在采购订货信息表格中选中要删除的内容，再单击页面中的“删除”按钮，即可实现删除操作。实现删除功能的具体实现步骤如下。

（1）定义删除数据 deleteStock()方法，该方法有一个 int 类型参数，用于指定要删除采购订货信息的编号，具体代码如下：

```
public void deleteStock(int id){
        conn = connection.getCon();                              //获取数据库连接
        String sql = "delete from tb_stock where id ="+id;        //定义删除数据 SQL 语句
        try {
                Statement statement = conn.createStatement();    //实例化 Statement 对象
                statement.executeUpdate(sql);                    //执行 SQL 语句
        } catch (SQLException e) {
                e.printStackTrace();
        }
}
```

（2）在“删除”按钮的单击事件中，获取用户选择的表格中选择的要删除的采购订货信息的编号，

再调用删除采购订货信息方法，具体代码如下：

```
JButton deleteButton = new JButton("删除");
deleteButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int row = table.getSelectedRow();                           //获取用户选择的表格的行号
        if (row < 0) {                                              //判断用户选择的行号是否大于 0
            JOptionPane.showMessageDialog(getParent(), "没有选择要删除的数据！",
                    "信息提示框", JOptionPane.INFORMATION_MESSAGE);
            return;                                                 //退出程序
        }
        String column = dm.getValueAt(row, 1).toString();           //获取用户选择的行的第一列数据
        dao.deleteStock(Integer.parseInt(column));                  //调用删除数据方法
        JOptionPane.showMessageDialog(getParent(), "数据删除成功！",
                "信息提示框", JOptionPane.INFORMATION_MESSAGE); //给出提示信息
    }
});
```

20.7　人员管理模块设计

20.7.1　人员管理模块概述

人员管理模块为超市管理员提供了管理超市内部员工的功能，人员管理模块涉及 4 张表，分别为部门信息表、职务信息表、员工基本信息表、员工详细信息表。人员管理窗体运行效果如图 20.16 所示。



图 20.16　人员管理窗体运行效果



20.7.2　使用触发器级联删除数据

本模块在保存员工信息时，使用了两张表，分别为员工基本信息表与员工详细信息表，这两个表中的数据是一一对应的，如果在员工基本表中删除数据后，对应的员工详细信息表中的数据也应该删除，因此可以通过创建 DELETE 触发器来实现。创建触发器要在数据库中实现，具体语法如下：

```
CREATE TRIGGER trigger_name
ON {table | view}
[WITH ENCRYPTION]
{
    {{FOR | AFTER | INSTEAD OF} {[INSERT] [,] [UPDATE]}
        [WITH APPEND]
        [NOT FOR REPLICATION]
        AS
        [{IF UPDATE (column)
           [{AND | OR} UPDATE (column)]
               [...n]
        | IF (COLUMNS_UPDATED() {bitwise_operator} updated_bitmask)
               {comparison_operator} column_bitmask [...n]
      }]
        sql_statement [...n]
   }
}
```

参数说明如表 20.7 所示。

表 20.7　CREATE TRIGGER 函数的参数说明

参　　数	说　　明
trigger_name	所要创建的触发器的名称
table|view	指创建触发器所在的表或视图，也可以称为触发器表或触发器视图
AFTER	指定触发器只有在完成指定的所有 SQL 语句之后才会被触发
AS	触发器要执行的操作
sql_statement	触发器的条件或操作。触发器条件指定其他准则，以确定 DELETE、INSERT 或 UPDATE 语句是否导致执行触发器

例如，本系统中员工基本信息表上创建触发器，实现删除指定的员工信息时，对应员工详细信息表中的数据也将删除。具体代码如下：

```
create trigger triGradeDelete on tb_basicMessage
for delete
 as
  declare @id varchar(10)
  select @id = id from deleted
  delete from tb_contact where tb_contact.id = @id
```

20.7.3 显示查询条件实现过程

本系统中将查询员工信息的条件以列表的形式给出，其中部门列表中的数据是从部门信息表中查询并显示在窗体中的，当用户选择了要查询员工的部门，系统会将该部门中的所有员工名称都显示在姓名列表中。人员管理模块的查询条件设计效果如图 20.17 所示。

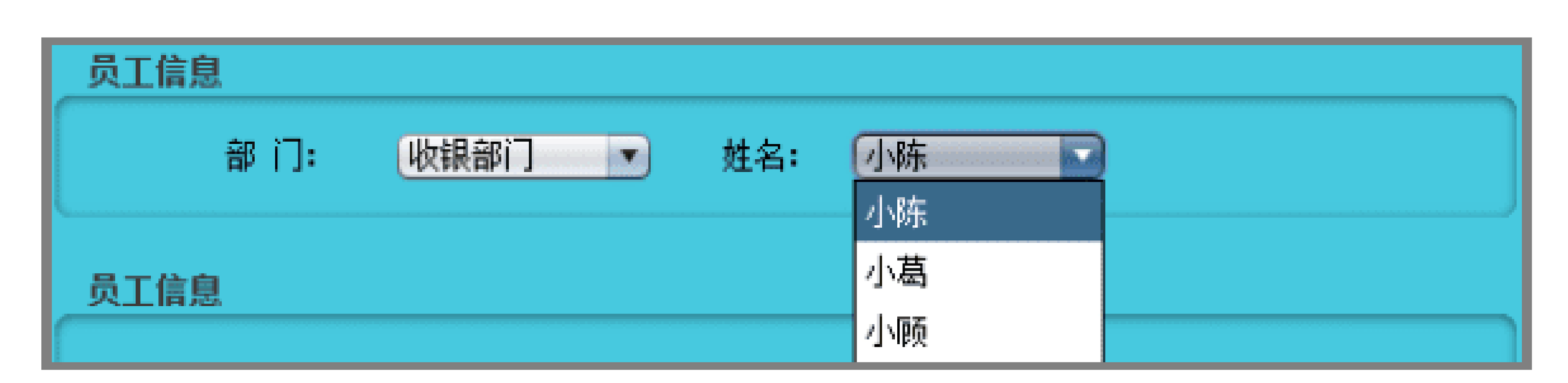


图 20.17　显示查询条件设计效果

下面详细地介绍显示查询条件的实现过程。

（1）定义查询部门信息表中所有数据方法 selectDept()，该方法将查询结果以 List 形式返回，具体代码如下：

```
public List selectDept() {
	List list = new ArrayList<Dept>();                    //定义 List 集合对象
	conn = connection.getCon();                           //获取数据库连接
	try {
		Statement statement = conn.createStatement();     //获取 Statement 方法
		ResultSet rest = statement.executeQuery("select * from tb_dept");
		//执行查询语句获取查询结果集
		while (rest.next()) {                             //循环遍历查询结果集
			Dept dept = new Dept();
			dept.setId(rest.getInt(1));                   //应用查询结果设置对象属性
			dept.setdName(rest.getString(2));
			dept.setPrincipal(rest.getString(3));
			dept.setBewrite(rest.getString(4));
			list.add(dept);                               //将对象添加到集合中
		}
	} catch (SQLException e) {
		e.printStackTrace();
	}
	return list;
}
```

（2）在人员管理窗体中，调用查询所有部门信息方法，并将查询出的结果显示在窗体中。具体代码如下：

```
List list = dao.selectDept();                             //调用查询所有部门信息方法
String dName[] = new String[list.size() + 1];             //根据查询结果创建字符串数组对象
dName[0] = "";
for (int i = 0; i < list.size(); i++) {                   //循环遍历查询结果集
	Dept dept = (Dept) list.get(i);
	dName[i + 1] = dept.getdName();                       //获取查询结果中部门名称
```



```
}
final JComboBox dNamecomboBox = new JComboBox(dName);        //实例化下拉列表对象
```

（3）定义查询指定部门中所有员工信息方法 selectBasicMessageByDept()，该方法将查询结果以 List 形式返回，具体代码如下：

```
public List selectBasicMessageByDept(int dept) {
    conn = connection.getCon();                              //获取数据库连接
    List list = new ArrayList<String>();                     //定义保存查询结果的集合对象
    try {
        Statement statement = conn.createStatement();        //实例化 Statement 对象
        String sql = "select name from tb_basicMessage where dept = " + dept +"";
                                                             //定义按照部门名称查询员工信息方法
        ResultSet rest = statement.executeQuery(sql);        //执行查询语句获取查询结果集
        while (rest.next()) {                                //循环遍历查询结果集
            list.add(rest.getString(1));                     //将查询信息保存到集合中
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;                                             //返回查询集合
}
```

（4）在“部门”下拉列表框中添加监听事件，实现当用户在“部门”下拉列表框中更改部门名称时，“姓名”下拉列表框中的内容也随之更新，具体代码如下：

```
final JComboBox dNamecomboBox = new JComboBox(dName);        //实例化下拉列表对象
    dNamecomboBox.addActionListener((new ActionListener() {  //添加下拉列表监听事件
            @Override
            public void actionPerformed(ActionEvent e) {
                String dName = dNamecomboBox.getSelectedItem().toString();
                //获取用户选择的部门名称
                DeptDao deptDao = new DeptDao();             //定义保存有操作数据库类对象
                int id = deptDao.selectDeptIdByName(dName);  //调用获取部门编号方法
                List<String> listName = perdao.selectBasicMessageByDept(id);
                //调用按部门编号查询所有员工信息方法
                for (int i = 0; i < listName.size(); i++) {  //循环遍历查询结果集
                    pNameComboBox.addItem(listName.get(i));
                    //向“姓名”下拉列表框中添加元素
                }
                repaint();
        }
}));
```

20.7.4 显示员工基本信息实现过程

当用户选择了要查询的员工后，单击“员工信息”列表中的“基本信息”列表项后，系统会将该

员工的基本信息显示在窗体中，运行结果如图 20.18 所示。

图 20.18　员工基本信息

下面介绍具体的实现过程。

（1）本模块的员工基本信息是通过员工部门信息与员工姓名查询出来的，首先编写按照部门名称和员工信息查询员工基本信息方法，具体代码如下：

```
public BasicMessage selectBNameById(String dept,String name) {
        conn = connection.getCon();                              //获取数据库连接
        BasicMessage message = new BasicMessage();               //创建与数据表对应的 JavaBean 对象
        try {
                Statement statement = conn.createStatement();
                String sql = "select * from tb_basicMessage where name = '"+name+"' and dept = (select id from
tb_dept" +" where dName = '"+dept+"')";                          //定义查询数据 SQL 语句
                ResultSet rest = statement.executeQuery(sql);  //执行查询语句
                while (rest.next()) {                            //循环遍历查询结果集
                        message.setId(rest.getInt(1));           //应用查询结果设置对象属性
                        message.setName(rest.getString(2));
                        message.setAge(rest.getInt(3));
                        message.setSex(rest.getString(4));
                        message.setDept(rest.getInt(5));
                        message.setHeadship(rest.getInt(6));
                }
        } catch (SQLException e) {
                e.printStackTrace();
        }
        return message;
}
```

（2）在员工信息窗体中，首先判断用户选择了合法的查询条件，再将用户选择要查询的员工的基本信息显示在窗体中，具体代码如下：

```
jlist.addListSelectionListener(new ListSelectionListener() {
        public void valueChanged(ListSelectionEvent e) {
                if (!e.getValueIsAdjusting()) {
                        String deptName = dNamecomboBox.getSelectedItem().toString();
                        //判断用户选择的查询条件
                        if(deptName.equals("")){
```



```
                JOptionPane.showMessageDialog(getParent(), "没有选择查询的员工！",
                        "信息提示框", JOptionPane.INFORMATION_MESSAGE);//提示信息
                return;
            }
            JList list = (JList) e.getSource();                       //获取事件源
            String value = (String) list.getSelectedValue();
            //获取列表选项并转换为字符串
            if(value.equals("基本信息")){                              //判断用户是否选择了"基本信息"
                String name = pNameComboBox.getSelectedItem().toString();
                //获取用户选择的员工姓名
                panel_1.remove(particular);                           //移除显示员工详细信息的面板
                panel_1.add(bpanel);
                bpanel.setBounds(140, 53, 409, 208);
                message = perdao.selectBNameById(deptName,name);
                //调用查询数据方法
                pId = message.getId();
                nameTextField.setText(message.getName());
                //设置员工基本系统中的组件内容
                ageTextField.setText(message.getAge()+"");
                String sex = message.getSex();
                if(sex.equals("男")){
                    manRadioButton.setSelected(true);
                    //设置窗体中"性别"单选按钮的显示内容
                }
                else{
                    wradioButton.setSelected(true);
                }
                int dept = message.getDept();
                Dept depts = dao.selectDepotById(dept);
                //按照部门编号查询员工所在部门信息
                deptField.setText(depts.getdName());                  //设置员工部门内容
                String hName =perdao.selectHeadshipById(message.getHeadship());
                headshipField.setText(hName);
                repaint();
            }
        }
});
```

20.7.5　添加员工信息实现过程

当用户单击如图 20.16 所示的员工信息窗体中的“添加”按钮后，将弹出添加员工信息窗体。添加员工信息由两部分组成，分别为添加员工基本信息与添加员工联系资料，添加员工信息窗体使用了选项卡面板，添加员工信息窗体运行效果如图 20.19 所示。

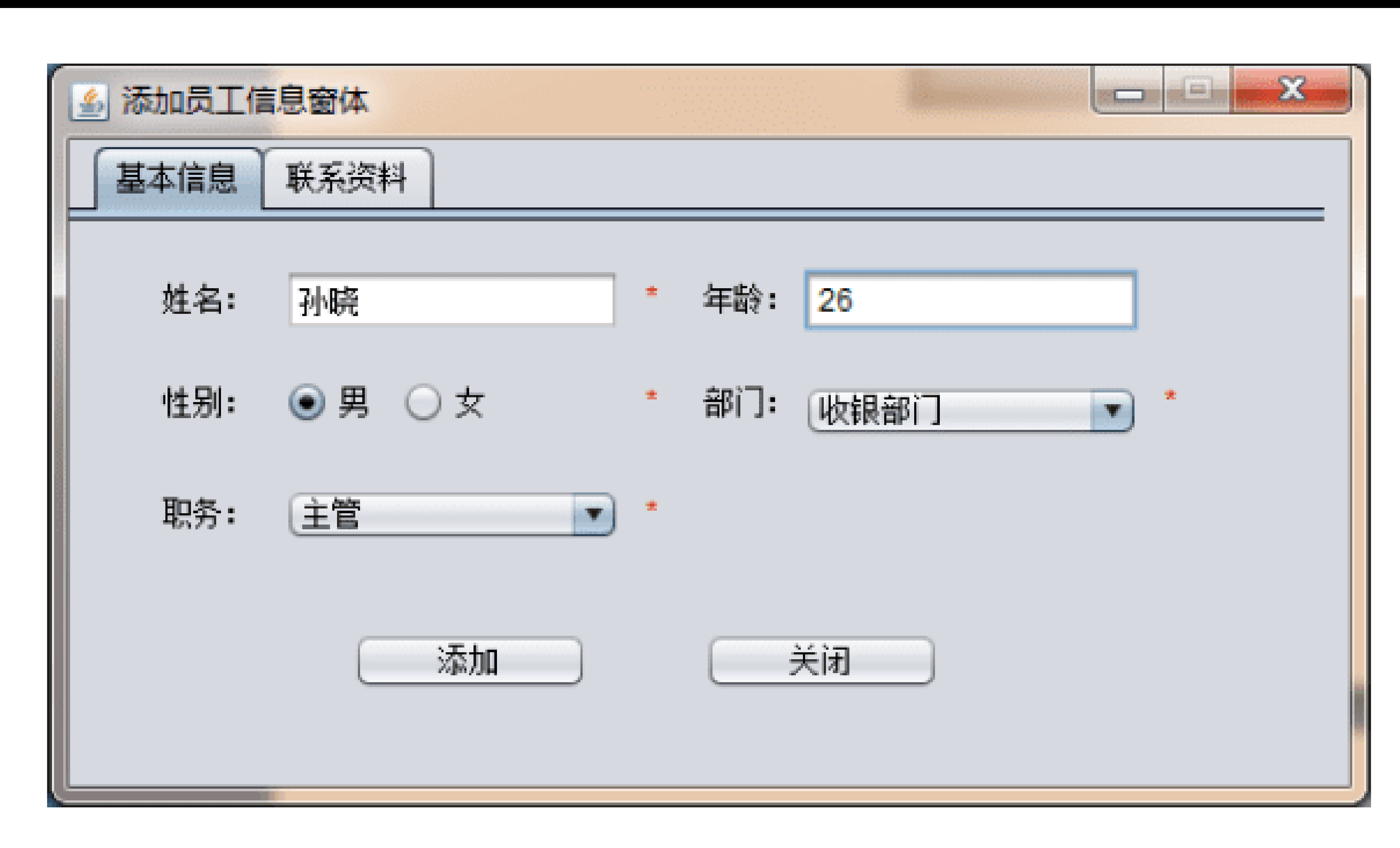


图 20.19　添加员工信息窗体

下面介绍具体的实现过程。

（1）定义向员工基本信息表中添加数据方法 insertBasicMessage()，该方法将与员工基本表对应的 JavaBean 对象 BasicMessage 作为参数，具体代码如下：

```
public void insertBasicMessage(BasicMessage message) {
	conn = connection.getCon();                                  //获取数据库连接
	try {
		PreparedStatement statement = conn
				.prepareStatement("insert into tb_basicMessage values(?,?,?,?,?)");
		//定义添加数据 SQL 语句
		statement.setString(1,message.getName());            //设置预处理语句参数值
		statement.setInt(2, message.getAge());
		statement.setString(3, message.getSex());
		statement.setInt(4, message.getDept());
		statement.setInt(5, message.getHeadship());
		statement.executeUpdate();                          //执行插入语句
	} catch (SQLException e) {
		e.printStackTrace();
	}
}
```

（2）定义向员工详细信息表中添加数据的方法 insertContact()，具体代码如下：

```
public void insertContact(Contact contact) {
	conn = connection.getCon();                                  //获取数据库连接
	try {
		PreparedStatement statement = conn
				.prepareStatement("insert into tb_contact values(?,?,?,?,?,?)");
		//定义插入数据 SQL 语句
		statement.setInt(1, contact.getHid());               //设置插入语句参数
		statement.setString(2, contact.getContact());
		statement.setString(3, contact.getOfficePhone());
		statement.setString(4, contact.getFax());
		statement.setString(5, contact.getEmail());
		statement.setString(6, contact.getFaddress());
		statement.executeUpdate();                          //执行插入语句
```

```
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

（3）由于在添加员工信息窗体中，部门名称以文字的形式显示给用户，而员工基本信息表中的部门字段存储的是部门表中的部门编号，因此要定义按部门名称查询部门编号方法 selectDeptByName()，具体代码如下：

```
public Dept selectDeptByName(String name) {
    conn = connection.getCon();                                 //获取数据库连接
    Dept dept = null;
    try {
        Statement statement = conn.createStatement();           //实例化 Statement 对象
        String sql = "select * from tb_dept where dName = '" + name +"'";//定义按部门名称查询部门信息 SQL 语句
        ResultSet rest = statement.executeQuery(sql);           //执行查询语句获取查询结果集
        while (rest.next()) {                                   //循环遍历查询结果集
            dept = new Dept();                                  //定义与部门表对应的 JavaBean 对象
            dept.setId(rest.getInt(1));                         //应用查询结果设置对象属性
            dept.setdName(rest.getString(2));
            dept.setPrincipal(rest.getString(3));
            dept.setBewrite(rest.getString(4));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return dept;                                                //返回 JavaBean 对象
}
```

（4）定义按照职位名称查询职务编号方法 selectIdByHeadship()，该方法以表示 String 对象为参数，将查询结果以 int 形式返回，具体代码如下：

```
public int selectIdByHeadship(String hName) {
    int id = 0;                                                 //定义保存查询结果的 int 对象
    conn = connection.getCon();                                 //获取数据库连接
    try {
        Statement statement = conn.createStatement();           //定义 Statement 对象
        String sql = "select id from tb_headship where headshipName = '" + hName+"'";
        //定义执行查询的 SQL 语句
        ResultSet rest = statement.executeQuery(sql);           //执行查询语句获取查询结果集
        while (rest.next()) {                                   //循环遍历查询结果集
            id = rest.getInt(1);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return id;
}
```

（5）在添加员工信息窗体中，要用户输入年龄信息的文本框只允许输入数字，可通过在年龄文本

框中添加键盘监听事件，实现当用户输入字母时不显示在文本框中，具体代码如下：

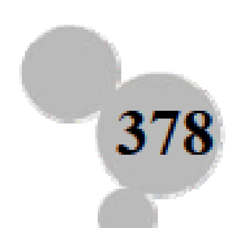
```
ageTextField = new JTextField();
ageTextField.addKeyListener(new KeyAdapter() {
    public void keyTyped(KeyEvent event) {                  //某键按下时调用的方法
        char ch = event.getKeyChar();                       //获取用户键入的字符
        if((ch<'0'||ch>'9')){                               //如果用户输入的信息不为数字
            event.consume();                                //不允许用户键入
        }
    }
});
```

（6）在“添加”按钮的监听事件中，首先判断用户是否输入合法的信息，如果输入合法，则实现将用户添加的信息保存到数据库中，具体代码如下：

```
JButton insertutton = new JButton("添加");
insertutton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String name = nameTextField.getText();              //获取用户添加的姓名信息
        String age = ageTextField.getText();                //获取用户添加的年龄信息
        String dept = deptComboBox.getSelectedItem().toString();
        //获取用户添加的部门信息
        String headship = headshipComboBox.getSelectedItem().toString();
        //获取用户添加的职务信息
        int id = dao.selectIdByHeadship(headship);          //调用根据职务名称查询职务编号方法
        if((name.equals(""))||(age.equals(""))){            //判断用户添加的信息是否为空
            JOptionPane.showMessageDialog(getContentPane(), "将带星号的信息填写完整！","信息提示框",
JOptionPane.INFORMATION_MESSAGE);                           //给出提示信息
            return;                                         //退出程序
        }
        int ageid = Integer.parseInt(age);                  //将用户添加的年龄信息转换为整型数据
        DeptDao deptDao = new DeptDao();                    //创建保存操作部门表数据方法
        Dept dpt = deptDao.selectDeptByName(dept);          //调用根据部门名称查询部门编号方法
        message.setName(name);                              //设置 JavaBean 对象名称属性
        message.setAge(ageid);
        message.setDept(dpt.getId());
        message.setHeadship(id);
        dao.insertBasicMessage(message);                    //调用向员工信息表中添加数据方法
        JOptionPane.showMessageDialog(getContentPane(), "将信息添加成功！",
                "信息提示框", JOptionPane.INFORMATION_MESSAGE);
    }
});
```

20.7.6　删除员工信息实现过程

当用户单击如图 20.16 所示的员工信息窗体中的“删除”按钮后，系统会将用户选择的员工删除，由于笔者在数据库中创建了触发器，因此当用户实现将员工基本信息表中的数据删除时，员工详细信息表中对应的数据也会被删除。



删除员工显示信息的具体实现过程如下。

（1）定义删除员工信息方法 deleteBasicMessage()，该方法以员工编号作为参数，具体代码如下：

```
public void deleteBasicMessage(int id){
	conn = connection.getCon();	//调用获取数据库连接方法
	String sql = "delete from tb_basicMessage where id ="+id;	//定义删除数据的 SQL 语句
	try {
		Statement statement = conn.createStatement();	//定义 Statement 方法
		statement.executeUpdate(sql);	//执行删除数据 SQL 语句
	} catch (SQLException e) {
		e.printStackTrace();
	}
}
```

（2）在“删除”按钮的单击事件中，实现删除员工基本信息，此时触发器 triGradeDelete 会自动执行，将对应的员工的详细信息也删除，具体代码如下：

```
JButton deleteButton = new JButton("删除");
deleteButton.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
			int n = JOptionPane.showConfirmDialog(getParent(),
					"确认正确吗？","确认对话框", JOptionPane.YES_NO_CANCEL_OPTION);
			if(n == JOptionPane.YES_OPTION){	//如果用户确认信息
				perdao.deleteBasicMessage(pId);	//调用删除数据方法
			}
	}
});
```

20.8　在 Eclipse 中实现程序打包

完成了简易腾宇超市管理系统的开发，接下来的工作就是对该系统进行打包并交付用户使用。下面就以在 Eclipse 中将应用程序打包成 JAR 文件为例，来讲解应用程序的打包过程。

将简易腾宇超市管理系统打包成 JAR 文件的步骤如下。

（1）首先编写 JAR 的清单文件，在清单文件中完成 JAR 文件的配置，如闪屏界面、主类名称、类路径等。在 Eclipse 的“包资源管理器”视图中的超市管理管理系统节点上单击鼠标右键，在弹出的快捷菜单中选择“新建”→“文件”命令，打开“新建文件”窗口，如图 20.20 所示，在“文件名”文本框中输入 MANIFEST.MF，单击“完成”按钮，完成 MANIFEST.MF 文件的建立。

（2）双击项目节点中的 MANIFEST.MF 文件，在打开 MANIFEST.MF 文件的编辑器中输入如下代码：

```
Manifest-Version: 1.0
SplashScreen-Image: res/sys_splash.jpg
Main-Class: com.mingrisoft.main.Enter
Class-Path: . lib/sqljdbc.jar
```

上面代码的第 1 行的 Manifest-Version 用于指定清单文件的版本。第 2 行的 SplashScreen-Image 用于指定闪屏界面所使用的图片资源，这里设置为 res/sys_splash.jpg，表示使用的是 res 包中的 sys_splash.jpg 文件。第 3 行的 Main-Class 用于定义 JAR 文件中的主类，这里设置为 com.mingrisoft. main.Enter。第 4 行的 Class-Path 用于设置程序执行时的类路径，运行程序所需的第三方类库，本应用程序使用的是 SQL Server 2014 数据库，所以需要把 SQL Server 数据库的驱动类添加到该类路径中。

注意

代码中的“:”必须要有一个空格字符作为分隔符。Class-Path 中的不同类库要使用空格分割。并且在清单文件的最后一行要有一个空行。

（3）保存 MANIFEST.MF 文件，在“包资源管理器”视图的“腾宇超市管理系统”节点上单击鼠标右键，在弹出的快捷菜单中选择“导出”命令，将打开“导出”窗口，如图 20.21 所示。选择 Java→“JAR 文件”节点，然后单击“下一步”按钮。




图 20.20 “新建文件”窗口



图 20.21 “导出”窗口

（4）在打开的“JAR 导出”窗口中的“JAR 文件”文本框中输入要生成的 JAR 文件的存放路径和文件名，这里输入“D:\超市管理系统\腾宇超市管理系统.jar”，如图 20.22 所示，单击“下一步”按钮。

（5）弹出“JAR 打包选项”界面，选中“导出带有编译错误的类文件”和“导出带有编译警告的类文件”复选框，如图 20.23 所示。因为类文件的编译警告信息不一定会导致程序无法运行，甚至有的警告信息并不影响项目要实现的业务逻辑，单击“下一步”按钮。

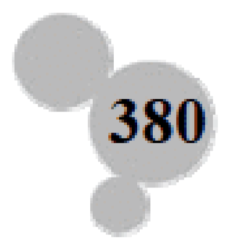




图 20.22　“JAR 导出”窗口



图 20.23　“JAR 打包选项”界面

（6）弹出“JAR 清单规范”界面，选中“从工作空间中使用现有清单”单选按钮，单击“清单文件”文本框右侧的“浏览”按钮，从打开的“选择清单”对话框中选择“腾宇超市管理系统”节点中的 MANIFEST.MF 清单文件，单击“确定”按钮。再单击“完成”按钮完成清单文件的选择，如图 20.24 所示。



图 20.24　“JAR 清单规范”界面

（7）打开“计算机”，双击 D 盘里的“超市管理系统”文件夹。在该文件夹中创建 lib 文件夹，如图 20.25 所示。把 JDBC 驱动类的 JAR 文件拷贝到 lib 文件夹中，如果客户端的 Java 环境安装正确的话，鼠标左键双击“超市管理系统.jar”文件就可以运行程序了。

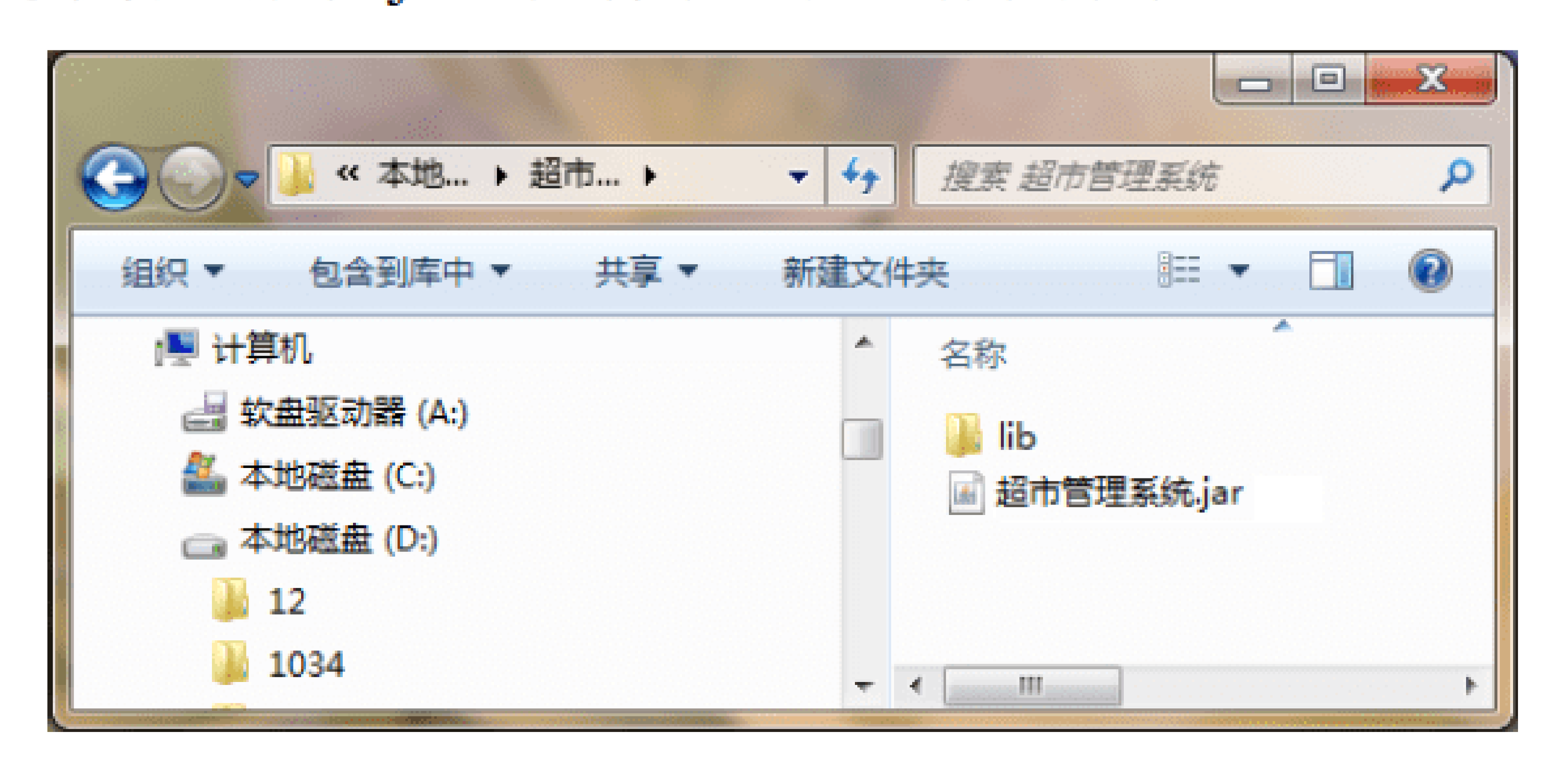


图 20.25　JAR 文件的存放地址

20.9　小　　结

超市管理系统从设计到开发应用了很多关键技术与项目开发技巧，这些技巧在开发中都是很关键的，下面将简略地介绍一下这些关键技术在实际中的用处，希望对读者的二次开发能有帮助。

（1）合理安排项目的包资源结构。例如，本项目中将所有操作数据库的类，都放在以 dao 命名的文件夹下，这样方便查找与后期的维护。

（2）合理地设计窗体的布局。开发的项目是要为用户使用的，因此设计良好的布局是十分关键的，这样可以给用户提供好的服务。

（3）面板的灵活使用。本系统主窗体的内容是随着用户选择的内容的不同而不断地更换的，这可以通过在窗体中更换不同的面板来实现。

（4）在表格中添加特殊内容。表格的默认内容是纯本文形式的，如果要添加特殊的内容要通过渲染器来实现。例如，本系统中在表格中添加按钮。

（5）合理地使用数据对象。数据对象在开发中是非常重要的，如触发器、视图、存储过程等。例如，本系统中在删除员工基本信息时使用了触发器。

第 21 章　学生成绩管理系统（Java+SQL Server 2014 实现）

随着教育的不断普及，各个学校的学生人数也越来越多。传统的管理方式已不能适应时代的发展。为了提高管理效率，减少学校开支，使用软件管理已成为必然。本章将开发一个学生成绩管理系统。通过本章的学习，可以掌握以下要点：

- ☑ Swing 控件的使用
- ☑ 内部窗体技术的使用
- ☑ JDBC 技术连接数据库

21.1　系 统 概 述

校园学生信息管理工作一直被视为校园管理中的一个瓶颈，积极寻求适应时代要求的校园学生信息管理模式已经成为当前校园管理工作的当务之急，学生信息管理是一门系统、普遍的科学，它是管理科学与教育科学中相互交融的综合性应用科学。学生信息管理范畴主要包括学籍管理、学科管理、课外活动管理、学生成绩管理、生活管理等。传统的人力管理模式既浪费校园人力，同时管理效果又不够明显，当将计算机管理系统深入校园学生信息管理工作时，学生信息管理工作中的数据信息被处理得更加精确，同时计算机管理为实际学生管理工作提供了强有力的数据信息，校方可以根据这些数据信息及时地对各项工作做出调整，使学生管理工作更加人性化。由于篇幅有限，本章将主要设计校园学生信息管理中的学生成绩管理系统。

21.2　系 统 分 析

21.2.1　需求分析

需求分析是系统项目开发的开端，经过与客户需求的沟通与协调，以及实际的调查与分析，本系统应该具有以下功能：

- ☑ 简单、友好的操作窗体，以方便管理员的日常管理工作。
- ☑ 整个系统的操作流程简单，易于操作。

☑ 完备的学生成绩管理功能。
☑ 全面的系统维护管理，方便系统日后维护工作。
☑ 强大的基础信息设置功能。

21.2.2 可行性研究

学生成绩管理系统是学生信息管理工作中的一部分，它一直以来是人们衡量学校优劣的一项重要指标，计算机管理系统深入学生成绩管理工作提高了对学生成绩管理工作的效率，更加有利于学校及时掌握学生的学习成绩、个人自然成长状况等一系列数据信息，通过这些实际数据，学校可以及时调整整个学校的学习管理工作。

21.3 系 统 设 计

21.3.1 系统目标

通过对学生成绩管理工作的调查与研究，要求本系统设计完成后将达到以下目标：
☑ 窗体界面设计友好、美观，方便管理员的日常操作。
☑ 基本信息的全面设置，数据录入方便、快捷。
☑ 数据检索功能强大、灵活，提高日常数据的管理工作。
☑ 具有良好的用户维护功能。
☑ 最大限度地实现系统易维护性和易操作性。
☑ 系统运行稳定、系统数据安全可靠。

21.3.2 系统功能结构

学生成绩管理系统的功能结构如图 21.1 所示。

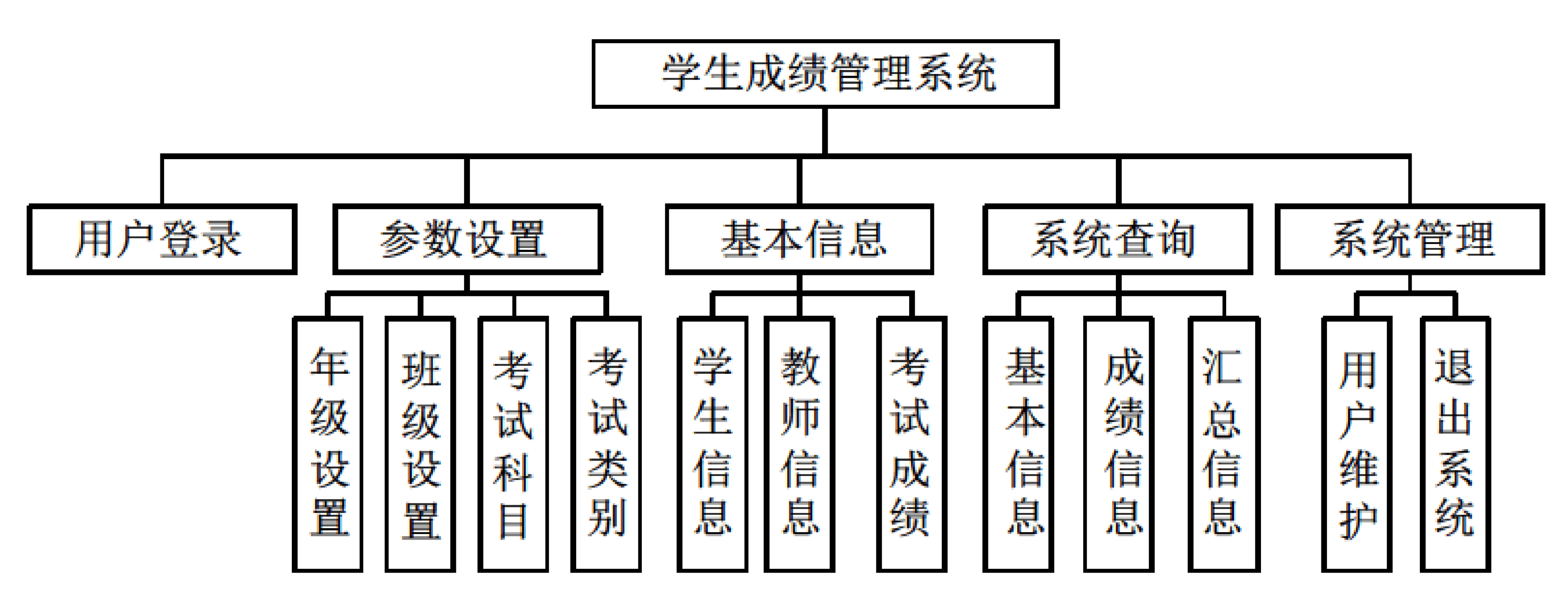
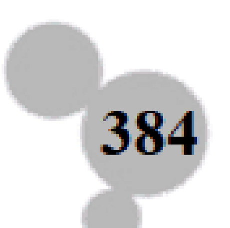


图 21.1 学生成绩管理系统的功能结构



21.3.3　系统预览

学生成绩管理系统由多个窗体组成，下面仅列出几个典型窗体，其他窗体参见资源包中的源程序。系统用户登录窗体的运行效果如图 21.2 所示，主要用于限制非法用户进入系统内部。

学生成绩管理系统主窗体的运行效果如图 21.3 所示，主要功能是调用执行本系统的所有功能。



图 21.2　系统用户登录窗体

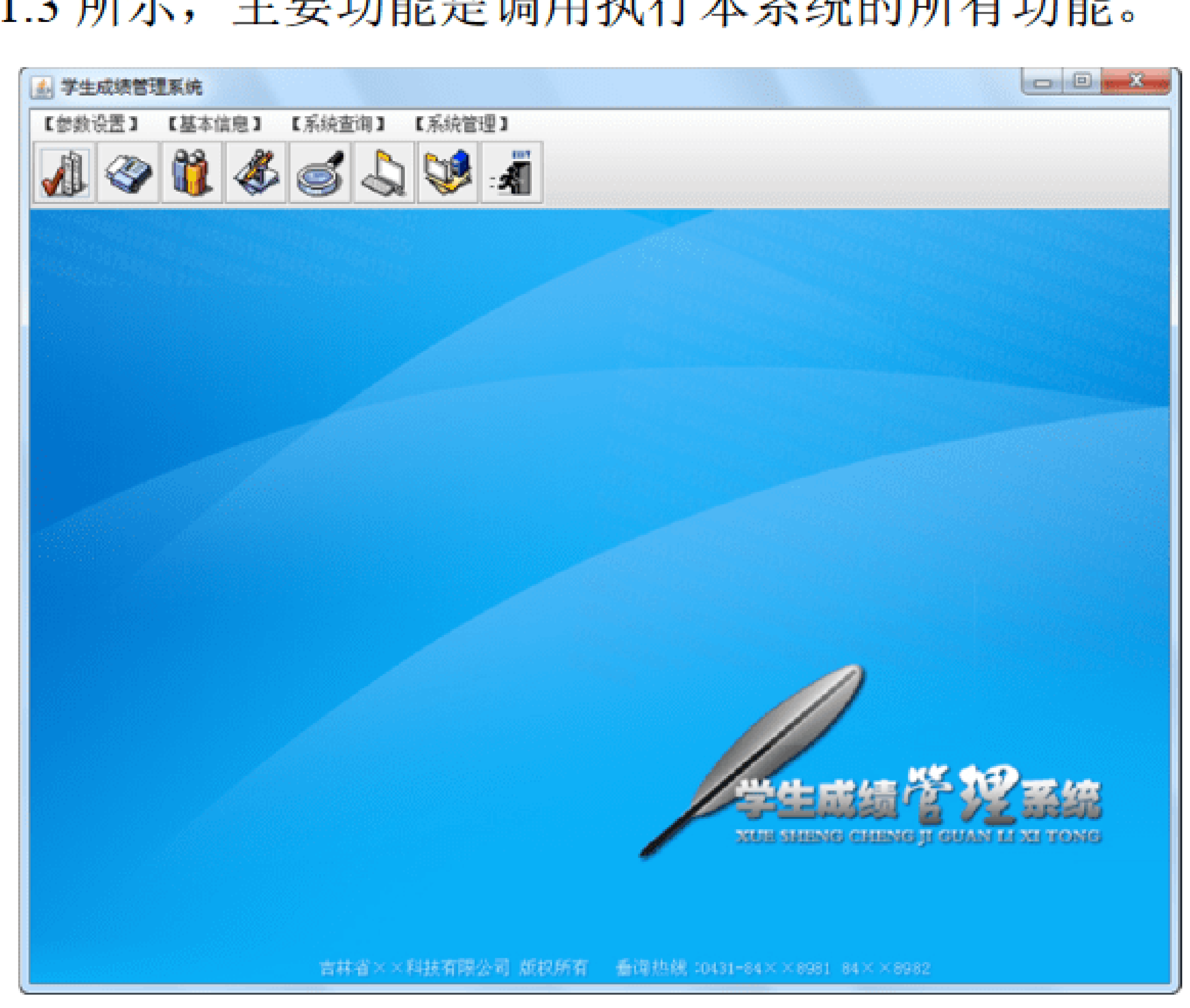


图 21.3　学生成绩管理系统主窗体

年级信息设置窗体的运行效果如图 21.4 所示，主要功能是对年级的信息进行增、删、改操作。

学生基本信息管理窗体的运行效果如图 21.5 所示，主要功能是对学生基本信息进行增、删、改操作。



图 21.4　年级信息设置窗体

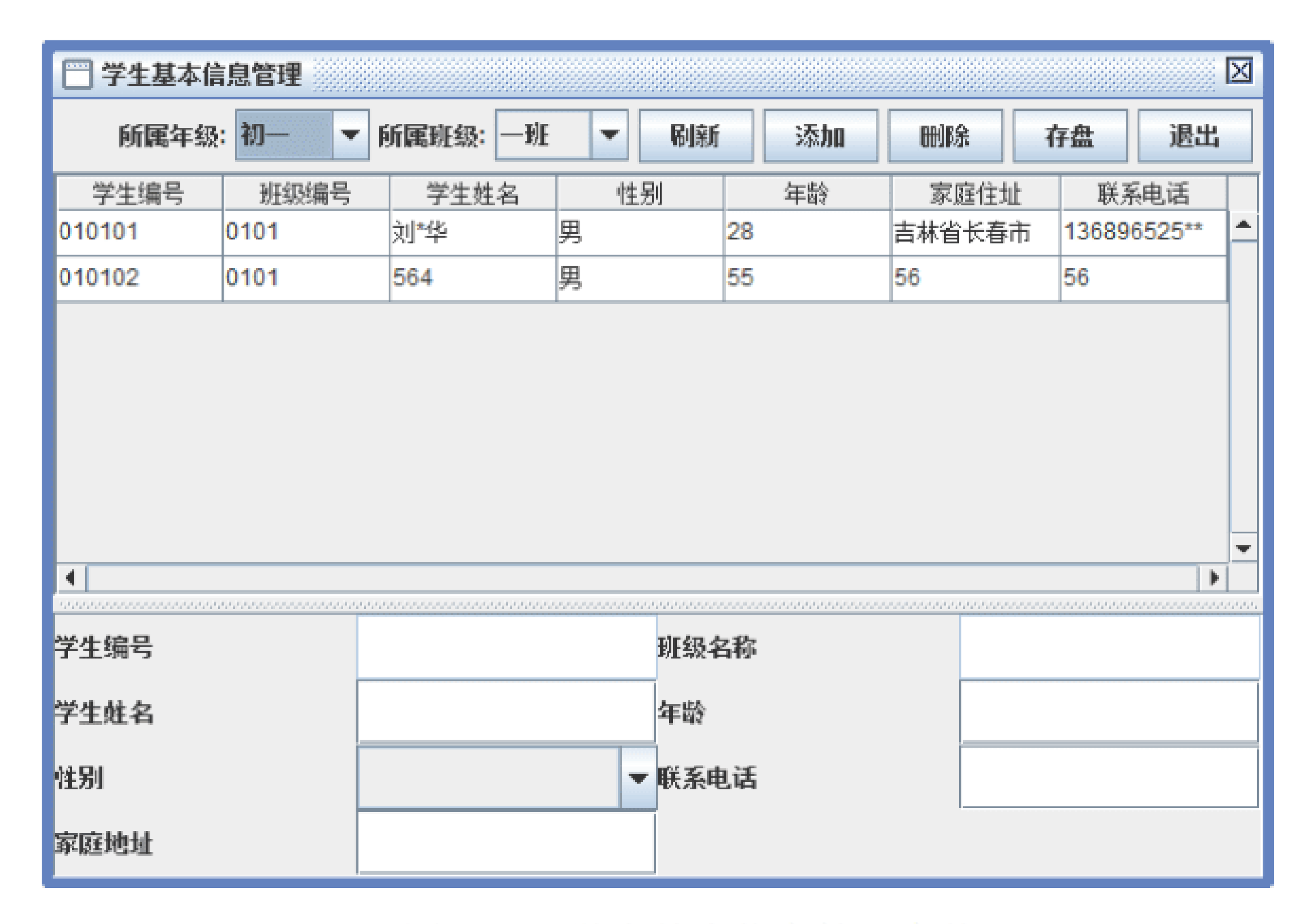
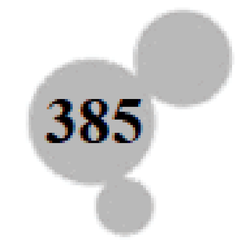


图 21.5　学生基本信息管理窗体

基本信息数据查询窗体的效果如图 21.6 所示，主要功能是查询学生的基本信息。



用户数据信息维护窗体的效果如图 21.7 所示，主要功能是完成用户信息的增加、修改和删除。



图 21.6　基本信息数据查询窗体

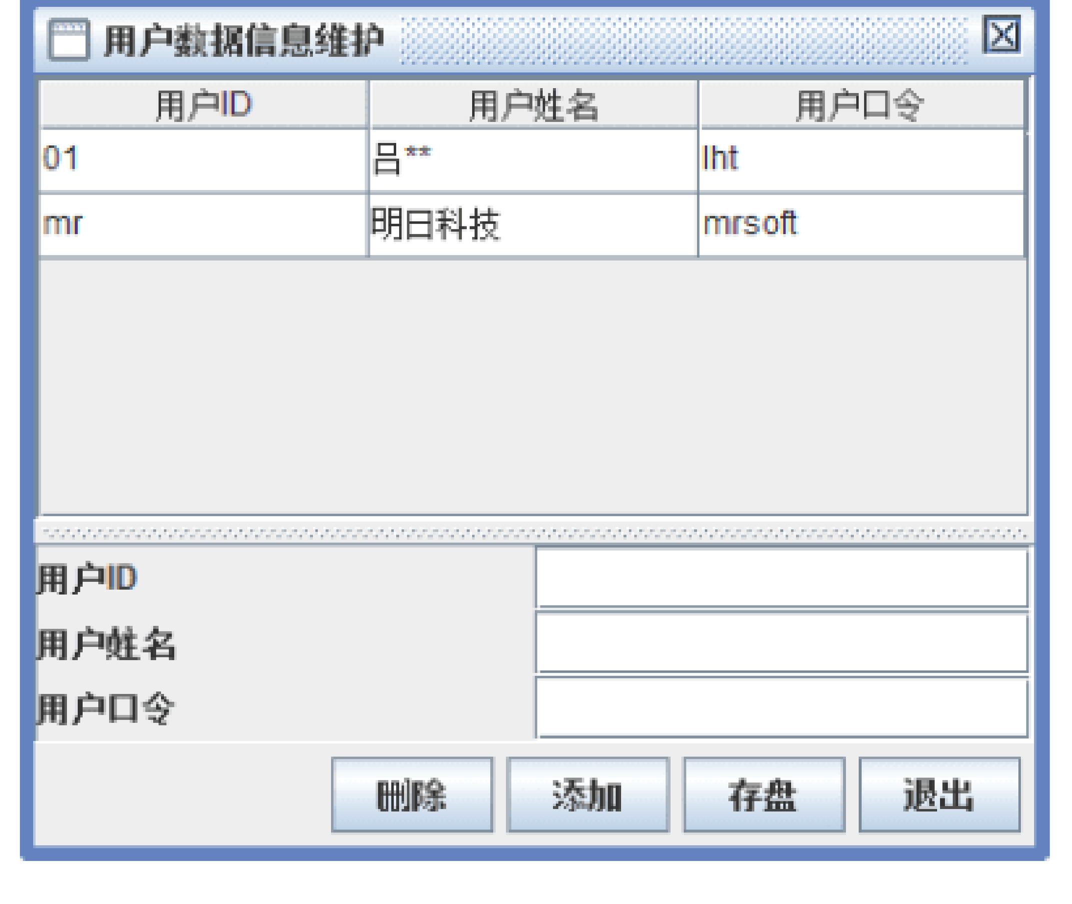


图 21.7　用户数据信息维护窗体

21.3.4　构建开发环境

在开发学生成绩管理系统时，需要具备下面的软件环境。

☑　操作系统：Windows 7 以上。

☑　Java 开发包：JDK 8 以上。

☑　数据库：SQL Server 2014。

21.3.5　文件夹组织结构

在进行系统开发前，需要规划文件夹组织结构，即建立多个文件夹，对各个功能模块进行划分，实现统一管理。这样做的好处为易于开发、管理和维护。本系统的文件夹组织结构如图 21.8 所示。

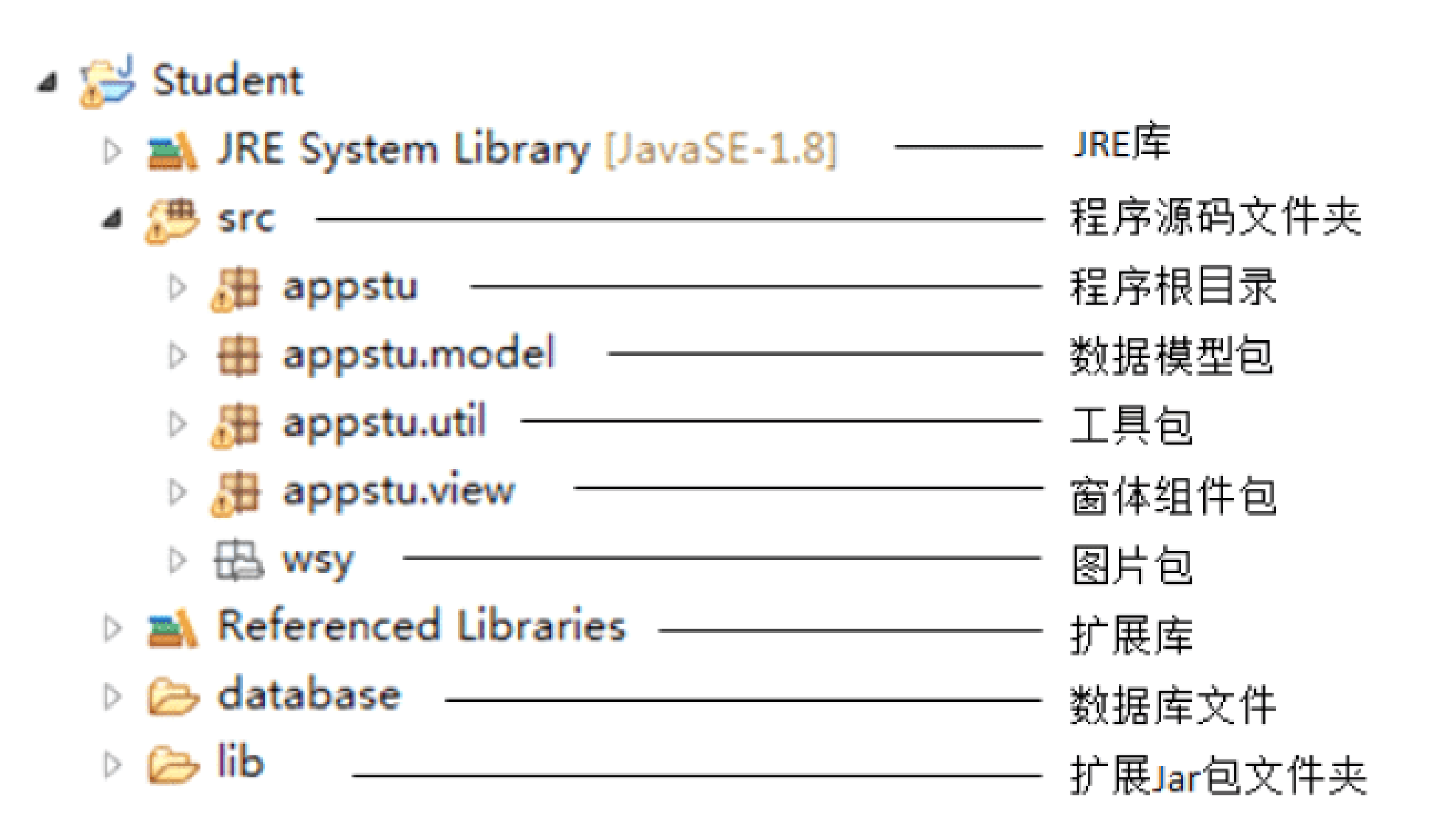
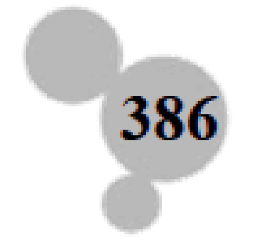


图 21.8　学生成绩管理系统文件夹组织结构



21.4　数据库设计

21.4.1　数据库分析

学生成绩管理系统主要用于管理整个学校的各方面信息，因此除了基本的学生信息表之外，还要设计教师信息表、班级信息表 1 年级信息表等。根据学生的学习成绩结构，设计科目表、考试种类表和考试科目成绩表等。

21.4.2　数据库概念设计

本系统数据库采用 SQL Server 2014 数据库，系统数据库名称为 DB_Student，共包含 8 张表。本系统数据表树状结构如图 21.9 所示，该数据表树状结构图包含系统所有数据表。

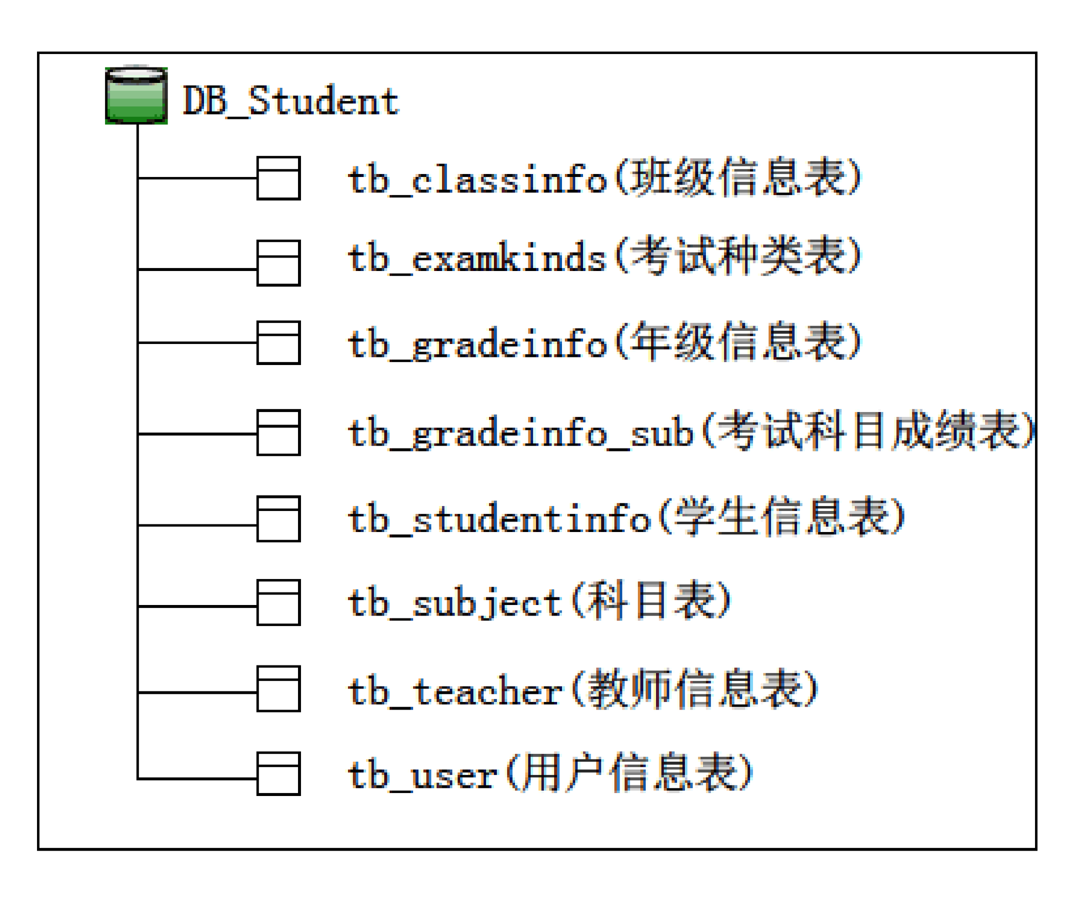


图 21.9　数据表树状结构

21.4.3　数据库逻辑结构设计

图 21.9 中各个表的详细说明如下。

☑　班级信息表

班级信息表 tb_classinfo 主要用于保存班级信息，其结构如表 21.1 所示。

表 21.1　tb_classinfo 结构

字段名称	数据类型	长　度	是否主键	描　述
classID	varchar	10	是	班级编号
gradeID	varchar	10		年级编号
className	varchar	20		班级名称

☑ 考试种类表

考试种类表（tb_examkinds）主要用来保存考试种类信息，其结构如表 21.2 所示。

表 21.2　tb_examkinds 结构

字段名称	数据类型	长　度	是否主键	描　述
kindID	varchar	20	是	考试类别编号
kindName	varchar	20		考试类别名称

☑ 年级信息表

年级信息表（tb_gradeinfo）用来保存年级信息，其结构如表 21.3 所示。

表 21.3　tb_gradeinfo 结构

字段名称	数据类型	长　度	是否主键	描　述
gradeID	varchar	10	是	年级编号
gradeName	varchar	20		年级名称

☑ 考试科目成绩表

考试科目成绩表（tb_gradeinfo_sub）用来保存考试科目成绩信息，其结构如表 21.4 所示。

表 21.4　tb_gradeinfo_sub 结构

字段名称	数据类型	长　度	是否主键	描　述
stuid	varchar	10	是	学生编号
stuname	varchar	50		学生姓名
kindID	varchar	10	是	考试类别编号
code	varchar	10	是	考试科目编号
grade	float	8		考试成绩
examdate	datetime	8		考试日期

☑ 学生信息表

学生信息表（tb_studentinfo）用来保存学生信息，其结构如表 21.5 所示。

表 21.5　tb_ studentinfo 结构

字段名称	数据类型	长　度	是否主键	描　述
stuid	varchar	10	是	学生编号
classID	varchar	10		班级编号
stuname	varchar	20		学生姓名
sex	varchar	10		学生性别
age	int	4		学生年龄
addr	varchar	50		家庭住址
phone	varchar	20		联系电话

☑　科目表

科目表（tb_subject）主要用来保存科目信息，其结构如表 21.6 所示。

表 21.6　tb_subject 结构

字段名称	数据类型	长　度	是否主键	描　述
code	varchar	10	是	科目编号
subject	varchar	40		科目名称

☑　教师信息表

教师信息表（tb_teacher）用于保存教师的相关信息，其结构如表 21.7 所示。

表 21.7　tb_teacher 结构

字段名称	数据类型	长　度	是否主键	描　述
teaid	varchar	10	是	教师编号
classID	varchar	10		班级编号
teaname	varchar	20		教师姓名
sex	varchar	10		教师性别
knowledge	varchar	20		教师职称
knowlevel	varchar	20		教师等级

☑　用户信息表

用户信息表（tb_user）主要用来保存用户的相关信息，其结构如表 21.8 所示。

表 21.8　tb_user 结构

字段名称	数据类型	长度	是否主键	描述
userid	varchar	50	是	用户编号
username	varchar	50		用户姓名
pass	varchar	50		用户口令

21.5　公共模块设计

实体类对象主要使用 JavaBean 来结构化后台数据表，完成对数据表的封装。在定义实体类时需要设置与数据表字段相对应的成员变量，并且需要为这些字段设置相应的 get 与 set 方法。

21.5.1　各种实体类的编写

在项目中通常会编写相应的实体类，下面笔者以学生实体类为例说明实体类的编写，它的步骤

如下。

（1）在 Eclipse 中，创建类 Obj_student.java，在类中创建与数据表 tb_studentinfo 字段相对应的成员变量。

（2）在 Eclipse 中的菜单栏中选择“源代码”/“生成 Getter 与 Setter”。

这样 Obj_student.java 实体类就创建完成了。它的代码如下：

```
public class Obj_student {
	private String stuid;                    // 定义学生编号变量
	private String classID;                  // 定义班级编号变量
	private String stuname;                  // 定义学生姓名变量
	private String sex;                      // 定义学生性别变量
	private int age;                         // 定义学生年龄变量
	private String address;                  // 定义家庭住址变量
	private String phone;                    // 定义联系电话变量
	public String getStuid() {
		return stuid;
	}
	public String getClassID() {
		return classID;
	}
	public String getStuname() {
		return stuname;
	}
	public String getSex() {
		return sex;
	}
	public int getAge() {
		return age;
	}
	public String getAddress() {
		return address;
	}
	public String getPhone() {
		return phone;
	}
	public void setStuid(String stuid) {
		this.stuid = stuid;
	}
	public void setClassID(String classID) {
		this.classID = classID;
	}
	public void setStuname(String stuname) {
		this.stuname = stuname;
	}
	public void setSex(String sex) {
```

```
            this.sex = sex;
      }
      public void setAge(int age) {
            this.age = age;
      }
      public void setAddress(String address) {
            this.address = address;
      }
      public void setPhone(String phone) {
            this.phone = phone;
      }
}
```

其他实体类的设计与学生实体类的设计相似，所不同的就是对应的后台表结构有所区别，读者在这里可以参考资源包中的源文件来完成。

21.5.2　操作数据库公共类的编写

1．连接数据库的公共类 CommonaJdbc.java

数据库连接在整个项目开发中占据着非常重要的位置，如果数据库连接失败，功能再强大的系统都不能运行。笔者在 appstu.util 包中建立类 CommonalJdbc.java 文件，在该文件中定义一个静态类型的类变量 connection 用来建立数据库的连接，这样在其他类中就可以直接访问这个变量了，其代码如下：

```
public class CommonaJdbc {
      public static Connection conection = null;
      public CommonaJdbc() {
            getCon();
      }
      private Connection getCon() {
            try {
                  Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
                  conection = DriverManager.getConnection(
                              "jdbc:sqlserver://localhost:1433;DatabaseName=DB_Student ", "sa", "123456");
            } catch (java.lang.ClassNotFoundException classnotfound) {
                  classnotfound.printStackTrace();
            } catch (java.sql.SQLException sql) {
                  new appstu.view.JF_view_error(sql.getMessage());
                  sql.printStackTrace();
            }
            return conection;
      }
}
```

2．操作数据库的公共类 JdbcAdapter.java

在 util 包下建立公共类 JdbcAdapter.java 文件，该类封装了对所有数据表的增加、修改、删除操作，

前台业务中的相应功能都是通过这个类来完成的，它的设计步骤如下。

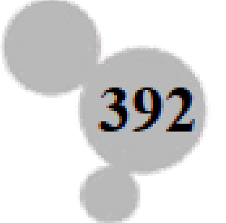
（1）该类通过在 21.5.1 节中设计的各种实体对象作为参数，进而执行类中的相应方法。为了保证数据操作的准确性，需要定义一个私有类方法 validateID 来完成数据的验证功能，这个方法首先通过数据表的主键判断数据表中是否存在这条数据：如果存在，则生成数据表的更新语句；如果不存在则生成表的添加语句。该方法的关键代码如下：

```
private boolean validateID(String id, String tname, String idvalue) {
    String sqlStr = null;
    sqlStr = "select count(*) from " + tname + " where " + id + " = '" + idvalue + "'";   // 定义 SQL 语句
    try {
        con = CommonaJdbc.conection;                                  // 获取数据库连接
        pstmt = con.prepareStatement(sqlStr);                         // 获取 PreparedStatement 实例
        java.sql.ResultSet rs = null;                                 // 获取 ResultSet 实例
        rs = pstmt.executeQuery();                                    // 执行 SQL 语句
        if (rs.next()) {
            if (rs.getInt(1) > 0)                                     // 如果数据表中有值
                return true;                                          // 返回 true 值
        }
    } catch (java.sql.SQLException sql) {                             // 如果产生异常
        sql.printStackTrace();                                        // 输出异常
        return false;                                                 // 返回 false 值
    }
    return false;                                                     // 返回 false 值
}
```

（2）定义一个私有类方法 AdapterObject 用来执行数据表的所有操作，方法参数为生成的 SQL 语句。该方法的关键代码如下：

```
private boolean AdapterObject(String sqlState) {
    boolean flag = false;
    try {
        con = CommonaJdbc.conection;                                  // 获取数据库连接
        pstmt = con.prepareStatement(sqlState);                       // 获取 PreparedStatement 实例
        pstmt.execute();                                              // 执行该 SQL 语句
        flag = true;                                                  // 将标识量修改为 true
        JOptionPane.showMessageDialog(null, infoStr + "数据成功!!!", "系统提示",
                        JOptionPane.INFORMATION_MESSAGE);             // 弹出相应提示对话框
    } catch (java.sql.SQLException sql) {
        flag = false;
        sql.printStackTrace();
    }
    return flag;                                                      // 将标识量返回
}
```

（3）由于在这个类中封装了所有的表操作，其实现方法都是一样的，因此这里仅以操作学生表的



InsertOrUpdateObject 方法为例进行详细讲解，其他方法的编写读者参考资源包中的源代码。InsertOrUpdateObject 方法的关键代码如下：

```
public boolean InsertOrUpdateObject(Obj_student objstudent) {
    String sqlStatement = null;
    if (validateID("stuid", "tb_studentinfo", objstudent.getStuid())) {
        sqlStatement = "Update tb_studentinfo set stuid = '" + objstudent.getStuid() + "',classID = '"
                    + objstudent.getClassID() + "',stuname = '" + objstudent.getStuname() + "',sex = '"
                    + objstudent.getSex() + "',age = '" + objstudent.getAge() + "',addr ='"
                    + objstudent.getAddress() + "',phone = '"
                   + objstudent.getPhone() + "' where stuid = '" + objstudent.getStuid().trim() + "'";
        infoStr = "更新学生信息";
    } else {
        sqlStatement = "Insert tb_studentinfo(stuid,classid,stuname,sex,age,addr,phone) values ('"
                   + objstudent.getStuid() + "','" + objstudent.getClassID() + "','" + objstudent.getStuname() + "','"
                   + objstudent.getSex() + "','" + objstudent.getAge() + "','" + objstudent.getAddress() + "','"
                   + objstudent.getPhone() + "')";
        infoStr = "添加学生信息";
    }
    return AdapterObject(sqlStatement);
}
```

（4）定义一个公共方法 InsertOrUpdate_Obj_gradeinfo_sub，用来执行学生成绩存盘操作。这个方法的参数为学生成绩对象 Obj_gradeinfo_sub 数组变量，定义一个 String 类型变量 sqlStr，然后在循环体中调用 stmt 的 addBatch 方法，将 sqlStr 变量放入 Batch 中，最后执行 stmt 的 executeBatch 方法。其关键代码如下：

```
public boolean InsertOrUpdate_Obj_gradeinfo_sub(Obj_gradeinfo_sub[] object) {
    try {
        con = CommonaJdbc.conection;
        stmt = con.createStatement();
        for (int i = 0; i < object.length; i++) {
            String sqlStr = null;
            if (validateobjgradeinfo(object[i].getStuid(), object[i].getKindID(), object[i].getCode())) {
                sqlStr = "update tb_gradeinfo_sub set stuid = '" + object[i].getStuid() + "',stuname = '"
                        + object[i].getSutname() + "',kindID = '" + object[i].getKindID() + "',code = '"
                        + object[i].getCode() + "',grade = " + object[i].getGrade() + " ,examdate = '"
                        + object[i].getExamdate() + "' where stuid = '" + object[i].getStuid() + "' and kindID = '"
                        + object[i].getKindID() + "' and code = '" + object[i].getCode() + "'";

            } else {
                sqlStr = "insert   tb_gradeinfo_sub(stuid,stuname,kindID,code,grade,examdate)   values ('"
                        + object[i].getStuid() + "','" + object[i].getSutname() + "','" + object[i].getKindID() + "','"
                        + object[i].getCode() + "'," + object[i].getGrade() + " ,'" + object[i].getExamdate() + "')";
            }
```

```
                System.out.println("sqlStr = " + sqlStr);
                stmt.addBatch(sqlStr);
            }
            stmt.executeBatch();
            JOptionPane.showMessageDialog(null, "学生成绩数据存盘成功!!!", "系统提示",
                                        JOptionPane.INFORMATION_MESSAGE);
        } catch (java.sql.SQLException sqlerror) {
            new appstu.view.JF_view_error("错误信息为：" + sqlerror.getMessage());
            return false;
        }
        return true;
}
```

（5）定义一个公共方法 Delete_Obj_gradeinfo_sub，用来删除学生成绩。该方法的设计与方法 InsertOrUpdate_Obj_gradeinfo_sub 类似，通过循环控制来生成批处理语句，然后执行批处理命令，所不同的就是该方法所生成的语句是删除语句。Delete_Obj_gradeinfo_sub 方法的关键代码如下：

```
public boolean Delete_Obj_gradeinfo_sub(Obj_gradeinfo_sub[] object) {
    try {
        con = CommonaJdbc.conection;
        stmt = con.createStatement();
        for (int i = 0; i < object.length; i++) {
            String sqlStr = null;
            sqlStr = "Delete From tb_gradeinfo_sub   where stuid = '" + object[i].getStuid() + "' and kindID = '"
                    + object[i].getKindID() + "' and code = '"+ object[i].getCode() + "'";
            System.out.println("sqlStr = " + sqlStr);
            stmt.addBatch(sqlStr);
        }
        stmt.executeBatch();
        JOptionPane.showMessageDialog(null, "学生成绩数据数据删除成功!!!", "系统提示",
                            JOptionPane.INFORMATION_MESSAGE);
    } catch (java.sql.SQLException sqlerror) {
        new appstu.view.JF_view_error("错误信息为：" + sqlerror.getMessage());
        return false;
    }
    return true;
}
```

（6）定义一个删除数据表的公共类方法 DeleteObject，用来执行删除数据表的操作，其关键代码如下：

```
public boolean DeleteObject(String deleteSql) {
    infoStr = "删除";
    return AdapterObject(deleteSql);
}
```

3．检索数据的公共类 RetrieveObject.java

数据的检索功能在整个系统中占有重要位置，系统中的所有查询都是通过该公共类实现的，该公共类通过传递的查询语句调用相应的类方法，查询满足条件的数据或者数据集合。笔者在这个公共类中定义了 3 种不同的方法来满足系统的查询要求。

（1）定义一个类的公共方法 getObjectRow，用来检索一条满足条件的数据，该方法返回值类型为 Vector，其关键代码如下：

```
public Vector getObjectRow(String sqlStr) {
    Vector vdata = new Vector();                                    // 定义一个集合
    connection = CommonaJdbc.conection;                             // 获取一个数据库连接
    try {
        rs = connection.prepareStatement(sqlStr).executeQuery();    // 获取一个 ResultSet 实例
        rsmd = rs.getMetaData();                                    // 获取一个 ResultSetMetaData 实例
        while (rs.next()) {
            for (int i = 1; i <= rsmd.getColumnCount(); i++) {
                vdata.addElement(rs.getObject(i));                  // 将数据库结果集中的数据添加到集合中
            }
        }
    } catch (java.sql.SQLException sql) {
        sql.printStackTrace();
        return null;
    }
    return vdata;                                                   // 将集合返回
}
```

（2）定义一个类的公共方法 getTableCollection，用来检索满足条件的数据集合，该方法返回值类型为 Collection，其关键代码如下：

```
public Collection getTableCollection(String sqlStr) {
    Collection collection = new Vector();
    connection = CommonaJdbc.conection;
    try {
        rs = connection.prepareStatement(sqlStr).executeQuery();
        rsmd = rs.getMetaData();
        while (rs.next()) {
            Vector vdata = new Vector();
            for (int i = 1; i <= rsmd.getColumnCount(); i++) {
                vdata.addElement(rs.getObject(i));
            }
            collection.add(vdata);
        }
    } catch (java.sql.SQLException sql) {
        new appstu.view.JF_view_error("执行的 SQL 语句为:\n" + sqlStr + "\n 错误信息为：" + sql.getMessage());
        sql.printStackTrace();
        return null;
```

```
    }
    return collection;
}
```

（3）定义类方法 getTableModel 用来生成一个表格数据模型，该方法返回类型为 DefaultTable-Model，该方法中一个数组参数 name 用来生成表模型中的列名，方法 getTableModel 的关键代码如下：

```
public DefaultTableModel getTableModel(String[] name, String sqlStr) {
    Vector vname = new Vector();
    for (int i = 0; i < name.length; i++) {
        vname.addElement(name[i]);
    }
    DefaultTableModel tableModel = new DefaultTableModel(vname, 0); // 定义一个 DefaultTableModel 实例
    connection = CommonaJdbc.conection;
    try {
        rs = connection.prepareStatement(sqlStr).executeQuery();
        rsmd = rs.getMetaData();
        while (rs.next()) {
            Vector vdata = new Vector();
            for (int i = 1; i <= rsmd.getColumnCount(); i++) {
                vdata.addElement(rs.getObject(i));
            }
            tableModel.addRow(vdata);                    // 将集合添加到表格模型中
        }
    } catch (java.sql.SQLException sql) {
        sql.printStackTrace();
        return null;
    }
    return tableModel;                                   // 将表格模型实例返回
}
```

4．产生流水号的公共类 ProduceMaxBh.java

在 appstu.util 包下建立公共类文件 ProduceMaxBh.java，在这个类定义一个公共方法 getMaxBh，该方法用来生成一个最大的流水号码，首先通过参数来获得数据表中的最大号码，然后根据这个号码产生一个最大编号，其关键代码如下：

```
public String getMaxBh(String sqlStr, String whereID) {
    appstu.util.RetrieveObject reobject = new RetrieveObject();
    Vector vdata = null;
    Object obj = null;
    vdata = reobject.getObjectRow(sqlStr);
    obj = vdata.get(0);
    String maxbh = null, newbh = null;
    if (obj == null) {
        newbh = whereID + "01";
    } else {
```

```
            maxbh = String.valueOf(vdata.get(0));
            String subStr = maxbh.substring(maxbh.length() - 1, maxbh.length());
            subStr = String.valueOf(Integer.parseInt(subStr) + 1);
            if (subStr.length() == 1)
                subStr = "0" + subStr;
            newbh = whereID + subStr;
        }
        return newbh;
}
```

21.6　系统用户登录模块设计

21.6.1　系统用户登录模块概述

系统用户登录模块主要用来验证用户的登录信息，完成用户的登录功能。该模块的运行结果如图 21.10 所示。



图 21.10　系统用户登录窗体

21.6.2　系统用户登录模块技术分析

系统用户登录模块使用的主要技术是如何让窗体居中显示。为了让窗体居中显示，首先要获得显示器的大小。使用 Toolkit 类的 getScreenSize 方法可以获得屏幕的大小，该方法的声明如下：

```
public abstract Dimension getScreenSize() throws HeadlessException
```

但是 Toolkit 类是一个抽象类，不能够使用 new 获得其对象。该类中定义的 getDefaultToolkit()方法可以获得 Toolkit 类型的对象，该方法的声明如下：

```
public static Toolkit getDefaultToolkit()
```

在获得了屏幕的大小之后，通过简单的计算即可让窗体居中显示。

21.6.3　系统用户登录模块实现过程

1. 界面设计

登录窗体的界面设计比较简单，它的具体设计步骤如下：

（1）在 Eclipse 中的“包资源管理器”视图中选择项目，在项目的 src 文件夹上单击鼠标右键，选择“新建”/“其他”菜单项，在弹出“新建”对话框的“输入过滤文本”文本框中输入 JFrame，然后选择 WindowBuilder/Swing Designer/JFrame 节点。

（2）在 New JFrame 对话框中，输入包名为 appstu.view，类名为 JF_login，单击“完成”按钮。该文件继承 javax.swing 包下面的 JFrame 类，JFrame 类提供了一个包含标题、边框和其他平台专用修饰的顶层窗口。

（3）创建类完成后，单击编辑器左下角的 Designer 选项卡，打开 UI 设计器，设置布局管理器类型为 BorderLayout。

（4）在 Palette 控件托盘中选择 Swing Containers 区域中的 JPanel 按钮，将该控件拖曳到 contentPane 控件中，此时该 JPanel 默认放置在整个容器的中部，可以在 Properties 选项卡中的 constraints 对应的属性中修改该控件的布局。同时在 Palette 托盘中选择两个 JLabel、1 个 JTextFiled 和 1 个 JPasswordField 控件放置到 JPanel 容器中。设置这两个 JLabel 的 text 属性为“用户名”和“密码”。

（5）以相同的方式从 Palette 控件托盘中选择 1 个 JPanel 容器拖曳到 contentPane 控件中，设置该面板位于布局管理器的上部，然后在该面板中放置 1 个 JLabel 控件。然后再选择 1 个 JPanel 容器拖曳到 contentPane 控件中，使该面板位于布局管理器的下部，选择两个 JButton 控件放置在该面板中。

根据以上几个步骤就完成了整个用户登录的窗体设计，具体的 UI 设计器的 Property Editor 窗口效果图如图 21.11 所示。



图 21.11　JF_login 类中控件的名称

2. 代码设计

登录窗体的具体设置步骤如下：



（1）当用户输入用户名、密码后，按下 Enter 键，系统校验该用户是否存在。在公共方法 jTextField1_keyPressed 中，定义一个 String 类型变量 sqlSelect 用来生成 SQL 查询语句，然后再定义一个公共类 RetrieveObject 类型变量 retrieve，调用 retrieve 的 getObjectRow 方法，其参数为 sqlSelect，用来判断该用户是否存在。jTextField1_keyPressed 方法的关键代码如下：

```
public void jTextField1_keyPressed(KeyEvent keyEvent) {
    if (keyEvent.getKeyCode() == KeyEvent.VK_ENTER) {
        String sqlSelect = null;
        Vector vdata = null;
        // 根据用户的输入，查询在数据库中是否存在
        sqlSelect = "select username from tb_user where userid = '" + jTextField1.getText().trim() + "'";
        RetrieveObject retrieve = new RetrieveObject();
        vdata = retrieve.getObjectRow(sqlSelect);                    // 调用 getObjectRow 方法执行该 SQL 语句
        if (vdata.size() > 0) {
            jPasswordField1.requestFocus();                          // 焦点放置在密码框中
        } else {
            // 如果该用户名不存在，则弹出相应提示对话框
            JOptionPane.showMessageDialog(null, "输入的用户 ID 不存在，请重新输入!!!", "系统提示",
                                                                     JOptionPane.ERROR_MESSAGE);
            jTextField1.requestFocus();                              // 焦点放置在用户名文本框中
        }
    }
}
```

（2）如果用户存在，再输入对应的口令，输入的口令正确时，单击“登录”按钮，进入系统。公共方法 jBlogin_actionPerformed 的设计与 jTextField1_keyPressed 方法的设计相似，其关键代码如下：

```
public void jBlogin_actionPerformed(ActionEvent e) {
    if (jTextField1.getText().trim().length() == 0 || jPasswordField1.getPassword().length == 0) {
        JOptionPane.showMessageDialog(null, "用户密码不允许为空", "系统提示",
                                        JOptionPane.ERROR_MESSAGE);
        return;
    }
    String pass = null;
    pass = String.valueOf(jPasswordField1.getPassword());
    String sqlSelect = null;
    sqlSelect = "select count(*) from tb_user where userid = '" + jTextField1.getText().trim() + "' and pass = '"
            + pass + "'";
    Vector vdata = null;
    appstu.util.RetrieveObject retrieve = new appstu.util.RetrieveObject();
    vdata = retrieve.getObjectRow(sqlSelect);                              // 执行 SQL 语句
    if (Integer.parseInt(String.valueOf(vdata.get(0))) > 0) {              // 如果验证成功
        AppMain frame = new AppMain();                                     // 实例化系统主窗体
        this.setVisible(false);                                            // 设置该主窗体不可见
    } else {                                                               // 如果验证不成功
```

```
            JOptionPane.showMessageDialog(null, "输入的口令不正确,请重新输入!!!", "系统提示",
                                        JOptionPane.ERROR_MESSAGE);    // 弹出相应消息对话框
            jTextField1.setText(null);                                // 将用户名文本框置空
            jPasswordField1.setText(null);                            // 将密码文本框置空
            jTextField1.requestFocus();                               // 将焦点放置在用户名文本框中
            return;
        }
}
```

21.7 主窗体模块设计

21.7.1 主窗体模块概述

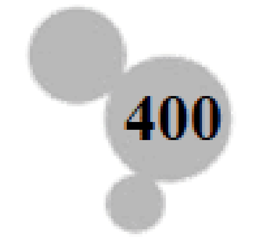
用户登录成功后，进入系统主界面，在主界面中主要完成对学生成绩信息的不同操作，其中包括各种参数的基本设置，学生/教师基本信息的录入、查询，成绩信息的录入、查询等功能。主窗体运行效果如图 21.12 所示。



图 21.12　学生成绩管理系统主窗体

21.7.2 主窗体模块技术分析

主窗体模块用到的主要技术是 JDesktopPane 类的使用。JDesktopPane 类用于创建多文档界面或虚拟桌面的容器。用户可创建 JInternalFrame 对象并将其添加到 JDesktopPane。JDesktopPane 扩展了 JLayeredPane，以管理可能的重叠内部窗体。它还维护了对 DesktopManager 实例的引用，这是由 UI 类



为当前的外观(L&F)所设置的。注意，JDesktopPane 不支持边界。

JDesktopPane 类通常用作 JInternalFrame 的父类，为 JInternalFrame 提供一个可插入的 DesktopManager 对象。特定于 L&F 的实现 installUI 负责正确设置 desktopManager 变量。JInternalFrame 的父类是 JDesktopPane 时，它应该将其大部分行为（关闭、调整大小等）委托给 desktopManager。

本模块使用了 JDesktopPane 类继承的 add 方法，它可以将指定的控件增加到指定的层次上，该方法的声明如下：

```
public Component add(Component comp,int index)。
```

- ☑ comp：要添加的控件。
- ☑ index：添加的控件的层次位置。

21.7.3　主窗体模块实现过程

1．界面设计

主界面的设计不是十分复杂，主要工作是在代码设计中完成。这里主要给出 UI 控件结构图，如图 21.13 所示。

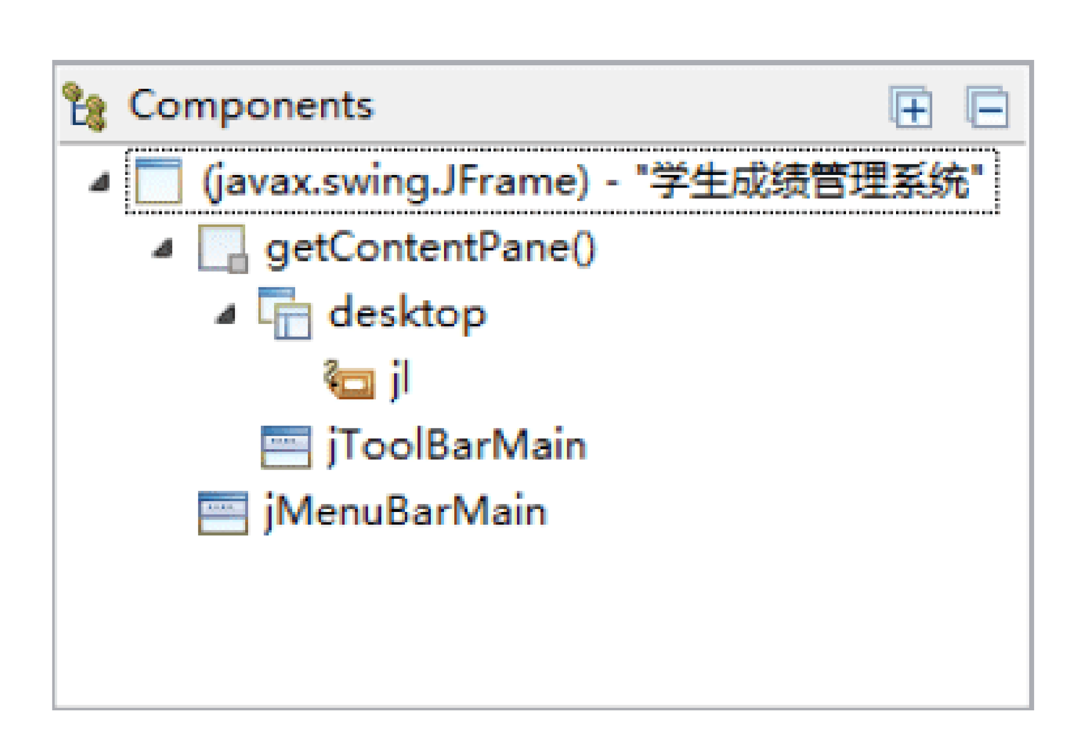


图 21.13　AppMain 类中控件的名称

2．代码设计

在主窗体中分别定义以下几个类的实例变量和公共方法：变量 JmenuBar 和 JToolBar（用来生成主界面中的主菜单和工具栏）、变量 MenuBarEvent（用来响应用户操作）和变量 JdesktopPane（用来生成放置控件的桌面面板）。定义完实例变量之后，开始定义创建主菜单的私有方法 BuildMenuBar 和创建工具栏的私有方法 BuildToolBar，其关键代码如下：

```
public class AppMain extends JFrame {
    // 省略部分代码
    public static JDesktopPane desktop = new JDesktopPane();
    MenuBarEvent _MenuBarEvent = new MenuBarEvent();              // 自定义事件类处理
    JMenuBar jMenuBarMain = new JMenuBar();                       // 定义界面中的主菜单控件
    JToolBar jToolBarMain = new JToolBar();                       // 定义界面中的工具栏控件
    private void BuildMenuBar() {                                 // 定义生成主菜单的公共方法
    }
    private void BuildToolBar() {                                 // 定义生成工具栏的公共方法
```

```
    }
    // 省略部分代码
}
```

下面分别详细讲述设置菜单栏与工具栏的方法。

（1）生成菜单的私有方法 BuildMenuBar 实现过程：首先定义菜单对象数组用来生成整个系统中的业务主菜单，然后定义主菜单中的子菜单项目，用来添加到主菜单中，为子菜单实现响应用户的单击的操作方法。关键代码如下：

```
private void BuildMenuBar() {
    JMenu[] _jMenu = { new JMenu("【参数设置】"), new JMenu("【基本信息】"), new JMenu("【系统查询】"),
                        new JMenu("【系统管理】") };
    JMenuItem[] _jMenuItem0 = { new JMenuItem("【年级设置】"), new JMenuItem("【班级设置】"),
                        new JMenuItem("【考试科目】"), new JMenuItem("【考试类别】") };
    String[] _jMenuItem0Name = { "sys_grade", "sys_class", "sys_subject", "sys_examkinds" };
    JMenuItem[] _jMenuItem1 = { new JMenuItem("【学生信息】"), new JMenuItem("【教师信息】"),
                        new JMenuItem("【考试成绩】") };
    String[] _jMenuItem1Name = { "JF_view_student", "JF_view_teacher", "JF_view_gradesub" };
    JMenuItem[] _jMenuItem2 = { new JMenuItem("【基本信息】"), new JMenuItem("【成绩信息】"),
                        new JMenuItem("【汇总查询】") };
    String[] _jMenuItem2Name = { "JF_view_query_jbqk", "JF_view_query_grade_mx", "JF_view_query_grade_hz" };
    JMenuItem[] _jMenuItem3 = { new JMenuItem("【用户维护】"), new JMenuItem("【系统退出】") };
    String[] _jMenuItem3Name = { "sys_user_modify", "JB_EXIT" };
    Font _MenuItemFont = new Font("宋体", 0, 12);
    for (int i = 0; i < _jMenu.length; i++) {
        _jMenu[i].setFont(_MenuItemFont);
        jMenuBarMain.add(_jMenu[i]);
    }
    for (int j = 0; j < _jMenuItem0.length; j++) {
        _jMenuItem0[j].setFont(_MenuItemFont);
        final String EventName1 = _jMenuItem0Name[j];
        _jMenuItem0[j].addActionListener(_MenuBarEvent);
        _jMenuItem0[j].addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                _MenuBarEvent.setEventName(EventName1);
            }
        });
        _jMenu[0].add(_jMenuItem0[j]);
        if (j == 1) {
            _jMenu[0].addSeparator();
        }
    }
    for (int j = 0; j < _jMenuItem1.length; j++) {
        _jMenuItem1[j].setFont(_MenuItemFont);
```

```
            final String EventName1 = _jMenuItem1Name[j];
            _jMenuItem1[j].addActionListener(_MenuBarEvent);
            _jMenuItem1[j].addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    _MenuBarEvent.setEventName(EventName1);
                }
            });
            _jMenu[1].add(_jMenuItem1[j]);
            if (j == 1) {
                _jMenu[1].addSeparator();
            }
        }
        for (int j = 0; j < _jMenuItem2.length; j++) {
            _jMenuItem2[j].setFont(_MenuItemFont);
            final String EventName2 = _jMenuItem2Name[j];
            _jMenuItem2[j].addActionListener(_MenuBarEvent);
            _jMenuItem2[j].addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    _MenuBarEvent.setEventName(EventName2);
                }
            });
            _jMenu[2].add(_jMenuItem2[j]);
            if ((j == 0)) {
                _jMenu[2].addSeparator();
            }
        }
        for (int j = 0; j < _jMenuItem3.length; j++) {
            _jMenuItem3[j].setFont(_MenuItemFont);
            final String EventName3 = _jMenuItem3Name[j];
            _jMenuItem3[j].addActionListener(_MenuBarEvent);
            _jMenuItem3[j].addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    _MenuBarEvent.setEventName(EventName3);
                }
            });
            _jMenu[3].add(_jMenuItem3[j]);
            if (j == 0) {
                _jMenu[3].addSeparator();
            }
        }
    }
```

（2）界面的主菜单设计完成之后，通过私有方法 BuildToolBar 进行工具栏的创建。定义 3 个 String

类型的局部数组变量，为工具栏上的按钮设置不同的数值，定义 JButton 控件，添加到实例变量 JToolBarMain 中。关键代码如下：

```
private void BuildToolBar() {
    String ImageName[] = { "科目设置.GIF", "班级设置.gif", "添加学生.gif", "录入成绩.GIF", "基本查询.GIF",
                           "成绩明细.GIF", "年级汇总.GIF", "系统退出.GIF" };
    String TipString[] = { "成绩科目设置", "学生班级设置", "添加学生", "录入考试成绩", "基本信息查询",
                           "考试成绩明细查询", "年级成绩汇总", "系统退出" };
    String ComandString[] = { "sys_subject", "sys_class", "JF_view_student", "JF_view_gradesub",
                    "JF_view_query_jbqk", "JF_view_query_grade_mx","JF_view_query_grade_hz", "JB_EXIT" };
    for (int i = 0; i < ComandString.length; i++) {
        JButton jb = new JButton();
        ImageIcon image = new ImageIcon(".\\images\\" + ImageName[i]);
        jb.setIcon(image);
        jb.setToolTipText(TipString[i]);
        jb.setActionCommand(ComandString[i]);
        jb.addActionListener(_MenuBarEvent);
        jToolBarMain.add(jb);
    }
}
```

21.8　班级信息设置模块设计

21.8.1　班级信息设置模块概述

班级信息设置用来维护班级的基本情况，包括对班级信息的添加、修改和删除等操作。在系统菜单栏中选择“参数设置”/“班级设置”选项，进入班级信息设置模块，其运行结果如图 21.14 所示。

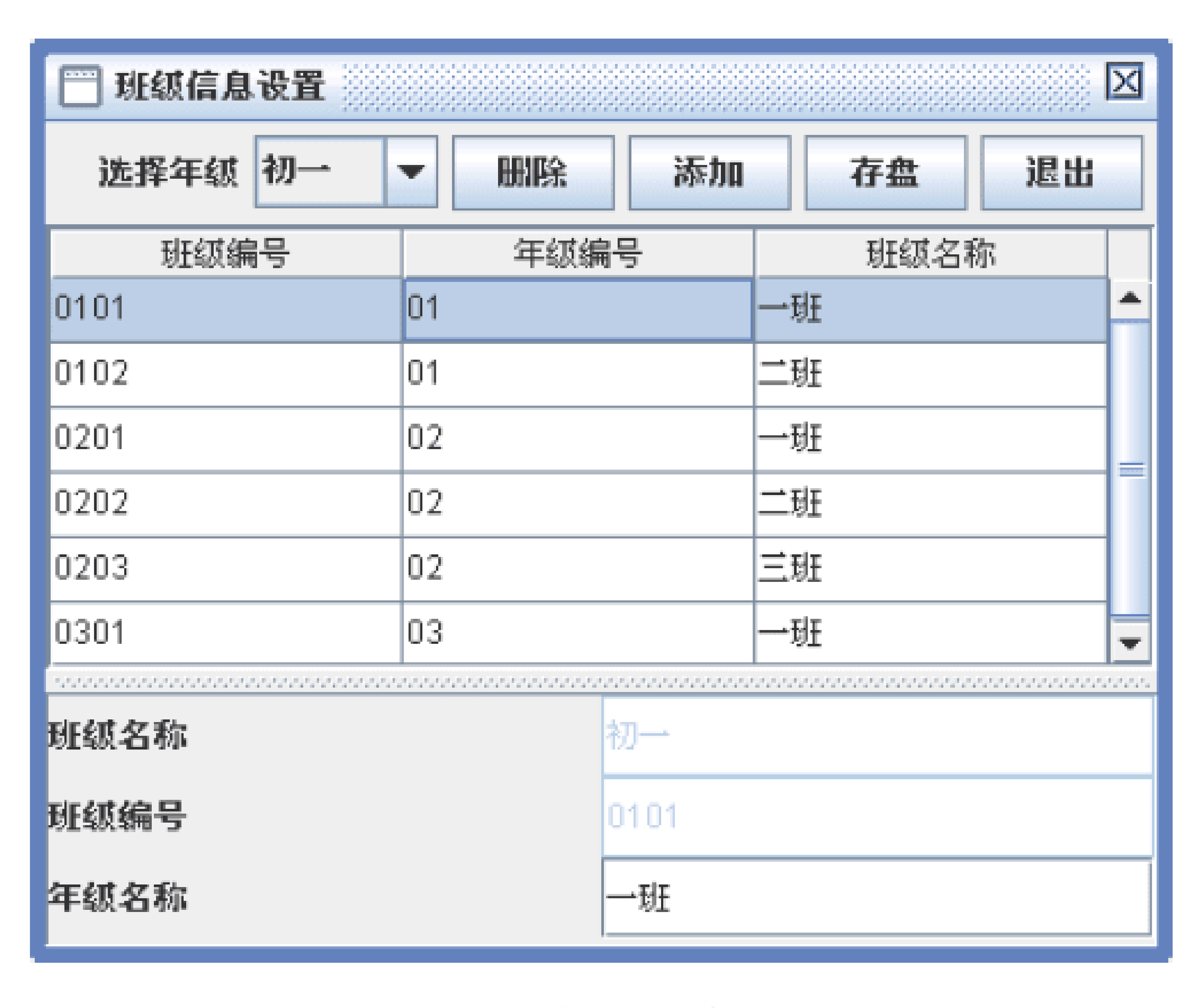


图 21.14　班级信息设置窗体运行效果图

21.8.2　班级信息设置模块技术分析

班级信息设置模块用到的主要技术是内部窗体的创建。通过继承 JInternalFrame 类，可以创建一个内部窗体。JInternalFrame 提供很多本机窗体功能的轻量级对象，这些功能包括拖动、关闭、变成图标、调整大小、标题显示和支持菜单栏。通常，可将 JInternalFrame 添加到 JDesktopPane 中。UI 将特定于外观的操作委托给由 JDesktopPane 维护的 DesktopManager 对象。

JInternalFrame 内容窗格是添加子控件的地方。为了方便地使用 add 方法及其变体，已经重写了 remove 和 setLayout，以在必要时将其转发到 contentPane。这意味着可以编写：

```
internalFrame.add(child);
```

子级将被添加到 contentPane。内容窗格实际上由 JRootPane 的实例管理，它还管理 layoutPane、glassPane 和内部窗体的可选菜单栏。

21.8.3　班级信息设置模块实现过程

1．界面设计

班级信息设置模块设计的窗体 UI 结构图如图 21.15 所示。

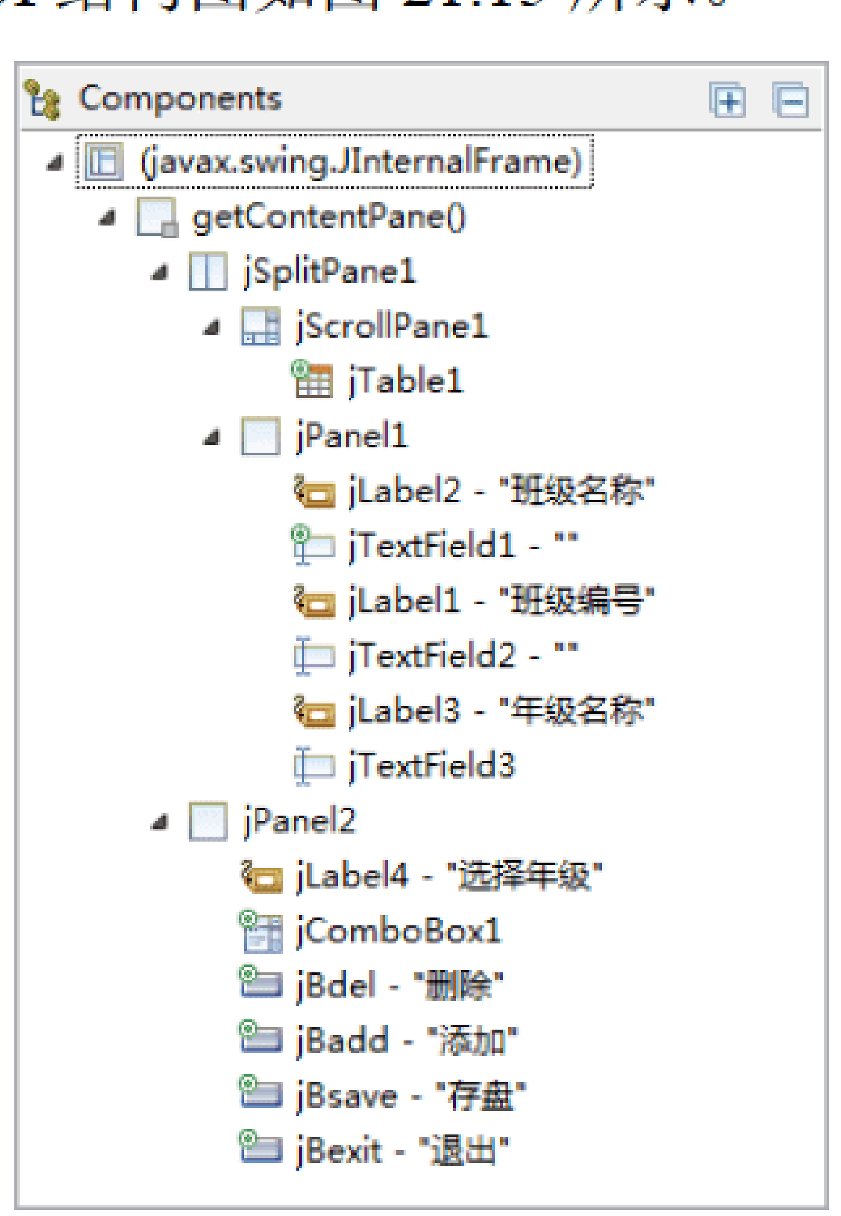


图 21.15　JF_view_sysset_class 类中控件的名称

2．代码设计

（1）通过调用上文中讲解的公共类 JdbcAdapter.java，完成对班级信息表 tb_classinfo 的相应操作。执行该模块程序，首先从数据表中检索出班级的基本信息，如果存在数据用户单击某一条数据之后可以对其进行修改、删除等操作。定义一个 boolean 实例变量 insertflag，用来标志操作数据库的类型，然后定义一个私有方法 buildTable，用来检索班级数据。其关键代码如下：

```
private void buildTable() {
```

```
    DefaultTableModel tablemodel = null;                    // 设置表格模型变量
    String[] name = { "班级编号", "年级编号", "班级名称" };   // 设置表头数组
    String sqlStr = "select * from tb_classinfo";           // 定义 SQL 语句
    appstu.util.RetrieveObject bdt = new appstu.util.RetrieveObject();
    tablemodel = bdt.getTableModel(name, sqlStr);            // 调用 getTableModel 方法获取一个表格模型实例
    jTable1.setModel(tablemodel);                            // 将表格模型放置在表格中
    jTable1.setRowHeight(24);                                // 设置表格的行高为 24
}
```

（2）单击“添加”按钮，用来增加一条新的数据信息。在公共方法 jBadd_actionPerformed 中定义局部字符串变量 sqlStr，用来生成查询最大编号的 SQL 语句，然后调用公共类 ProduceMaxBh 的 getMaxBh 方法生成最大编号，并显示在文本框中。其关键代码如下：

```
public void jBadd_actionPerformed(ActionEvent e) {
    // 获得年级名称
    if (jComboBox1.getItemCount() <= 0)
        return;
    int index = jComboBox1.getSelectedIndex();
    String gradeid = gradeID[index];
    String sqlStr = null, classid = null;
    sqlStr = "SELECT MAX(classID) FROM tb_classinfo where gradeID = '" + gradeid + "'";
    ProduceMaxBh pm = new appstu.util.ProduceMaxBh();
    System.out.println("我在方法 item 中" + sqlStr + "; index = " + index);
    classid = pm.getMaxBh(sqlStr, gradeid);
    jTextField1.setText(String.valueOf(jComboBox1.getSelectedItem()));
    jTextField2.setText(classid);
    jTextField3.setText("");
    jTextField3.requestFocus();
}
```

（3）用户单击表格上的某条数据后，程序会将这条数据填写到 jPanel2 面板上的相应控件上，以方便用户进行相应的操作，在公共方法 jTable1_mouseClicked 中定义一个 String 类型的局部变量 sqlStr，用来生成 SQL 查询语句，然后调用公共类 RetrieveObject 的 getObjectRow 方法，进行数据查询，如果找到数据则将该数据解析显示给用户，其关键代码如下：

```
public void jTable1_mouseClicked(MouseEvent e) {
    insertflag = false;
    String id = null;
    String sqlStr = null;
    int selectrow = 0;
    selectrow = jTable1.getSelectedRow();                   // 获取表格选定的行数
    if (selectrow < 0)
        return;                                             // 如果该行数小于 0，则返回
    id = jTable1.getValueAt(selectrow, 0).toString();       // 返回第 selectrow 行，第一列的单元格值
    // 根据编辑号内连接查询班级信息表与年级信息表中的基本信息
    sqlStr = "SELECT c.classID, d.gradeName, c.className FROM tb_classinfo c INNER JOIN " +
```

```
            " tb_gradeinfo d ON c.gradeID = d.gradeID where c.classID = '" + id + "'";
    Vector vdata = null;
    RetrieveObject retrive = new RetrieveObject();
    vdata = retrive.getObjectRow(sqlStr);                // 执行 SQL 语句返回一个集合
    jComboBox1.removeAllItems();
    jTextField1.setText(vdata.get(0).toString());
    jComboBox1.addItem(vdata.get(1));
    jTextField2.setText(vdata.get(2).toString());
}
```

（4）当对年级列表选择框 jComboBox1 进行赋值时，会自动触发 itemStateChanged 事件，为了解决对列表框的不同赋值操作（如浏览和删除），用到了实例变量 insertflag 进行判断。编写公共方法 jComboBox1_itemStateChanged 的关键代码如下：

```
public void jComboBox1_itemStateChanged(ItemEvent e) {
    if (insertflag) {
        String gradeID = null;
        gradeID = "0" + String.valueOf(jComboBox1.getSelectedIndex() + 1);
        ProduceMaxBh pm = new appstu.util.ProduceMaxBh();
        String sqlStr = null, classid = null;
        sqlStr = "SELECT MAX(classID) FROM tb_classinfo where gradeID = '" + gradeID + "'";
        classid = pm.getMaxBh(sqlStr, gradeID);
        jTextField1.setText(classid);
    } else {
        jTextField1.setText(String.valueOf(jTable1.getValueAt(jTable1.getSelectedRow(), 0)));
    }
}
```

（5）单击“删除”按钮，删除某一条班级数据信息。在公共方法 jBdel_actionPerformed 中定义字符串类型的局部变量 sqlDel，用来生成班级的删除语句，然后调用公共类的 JdbcAdapter 的 DeleteObject 方法。相关代码如下：

```
public void jBdel_actionPerformed(ActionEvent e) {
    int result = JOptionPane.showOptionDialog(null, "是否删除班级信息数据?", "系统提示",
        JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, new String[] {"是", "否" }, "否");
    if (result == JOptionPane.NO_OPTION)
        return;
    String sqlDel = "delete tb_classinfo where classID = '" + jTextField2.getText().trim() + "'";
    JdbcAdapter jdbcAdapter = new JdbcAdapter();
    if (jdbcAdapter.DeleteObject(sqlDel)) {
        jTextField1.setText("");
        jTextField2.setText("");
        jTextField3.setText("");
        buildTable();
    }
}
```

（6）单击“存盘”按钮，将数据保存在数据表中。在方法 jBsave_actionPerformed 中定义实体类

对象 Obj_classinfo，变量名为 objclassinfo，然后通过 set 方法为 objclassinfo 赋值，然后调用公共类 JdbcAdapter 的 InsertOrUpdateObject 方法，完成存盘操作，其参数为 objclassinfo。关键代码如下：

```
public void jBsave_actionPerformed(ActionEvent e) {
    int result = JOptionPane.showOptionDialog(null, "是否存盘班级信息数据?", "系统提示",
       JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, new String[] {"是", "否" }, "否");
    if (result == JOptionPane.NO_OPTION)
        return;
    int index = jComboBox1.getSelectedIndex();
    String gradeid = gradeID[index];
    appstu.model.Obj_classinfo objclassinfo = new appstu.model.Obj_classinfo();
    objclassinfo.setClassID(jTextField2.getText().trim());
    objclassinfo.setGradeID(gradeid);
    objclassinfo.setClassName(jTextField3.getText().trim());
    JdbcAdapter jdbcAdapter = new JdbcAdapter();
    if (jdbcAdapter.InsertOrUpdateObject(objclassinfo))
        buildTable();
}
```

21.9 学生基本信息管理模块设计

21.9.1 学生基本信息管理模块概述

学生基本信息管理模块用来管理学生基本信息，包括学生信息的添加、修改、删除、存盘等功能。单击菜单“基本信息”/“学生信息”选项，进入该模块，其运行结果如图 21.16 所示。

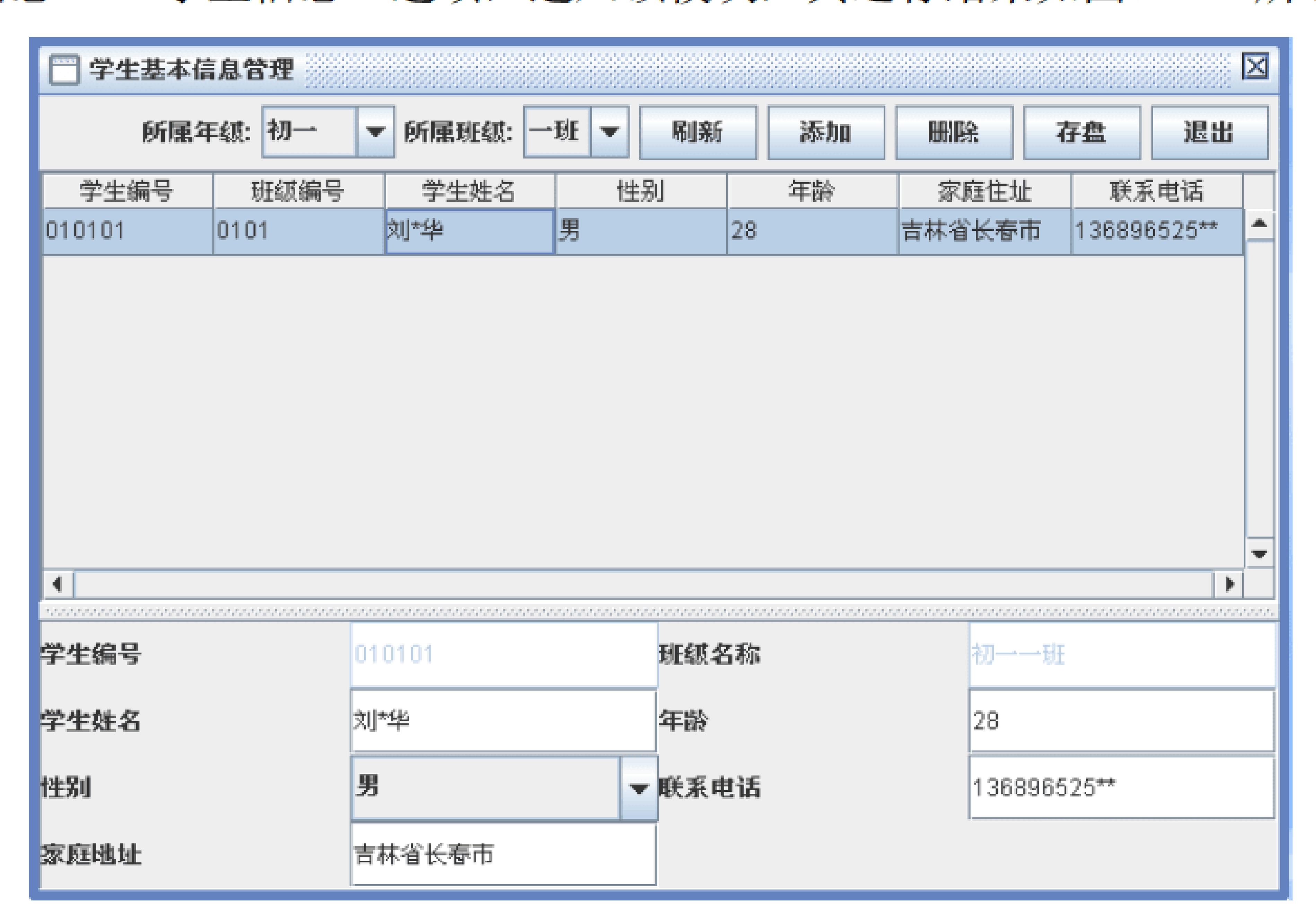


图 21.16　学生基本信息管理窗体



21.9.2　学生基本信息管理模块技术分析

学生基本信息管理模块中用到的主要技术是 JSplitPane 的使用。JSplitPane 用于分隔两个（只能两个）Component。两个 Component 图形化分隔以外观实现为基础，并且这两个 Component 可以由用户交互式调整大小。使用 JSplitPane.HORIZONTAL_SPLIT 可让分隔窗格中的两个 Component 从左到右排列，或者使用 JSplitPane.VERTICAL_SPLIT 使其从上到下排列。改变 Component 大小的首选方式是调用 setDividerLocation，其中 location 是新的 x 或 y 位置，具体取决于 JSplitPane 的方向。要将 Component 调整到其首选大小，可调用 resetToPreferredSizes。

当用户调整 Component 的大小时，Component 的最小大小用于确定 Component 能够设置的最大/最小位置。如果两个控件的最小大小大于分隔窗格的大小，则分隔条将不允许调整其大小。当用户调整分隔窗格大小时，新的空间以 resizeWeight 为基础在两个控件之间分配。默认情况下，值为 0 表示右边/底部的控件获得所有空间，而值为 1 表示左边/顶部的控件获得所有空间。

21.9.3　学生基本信息管理模块实现过程

1．界面设计

学生基本信息管理模块设计的窗体 UI 结构如图 21.17 和图 21.18 所示。

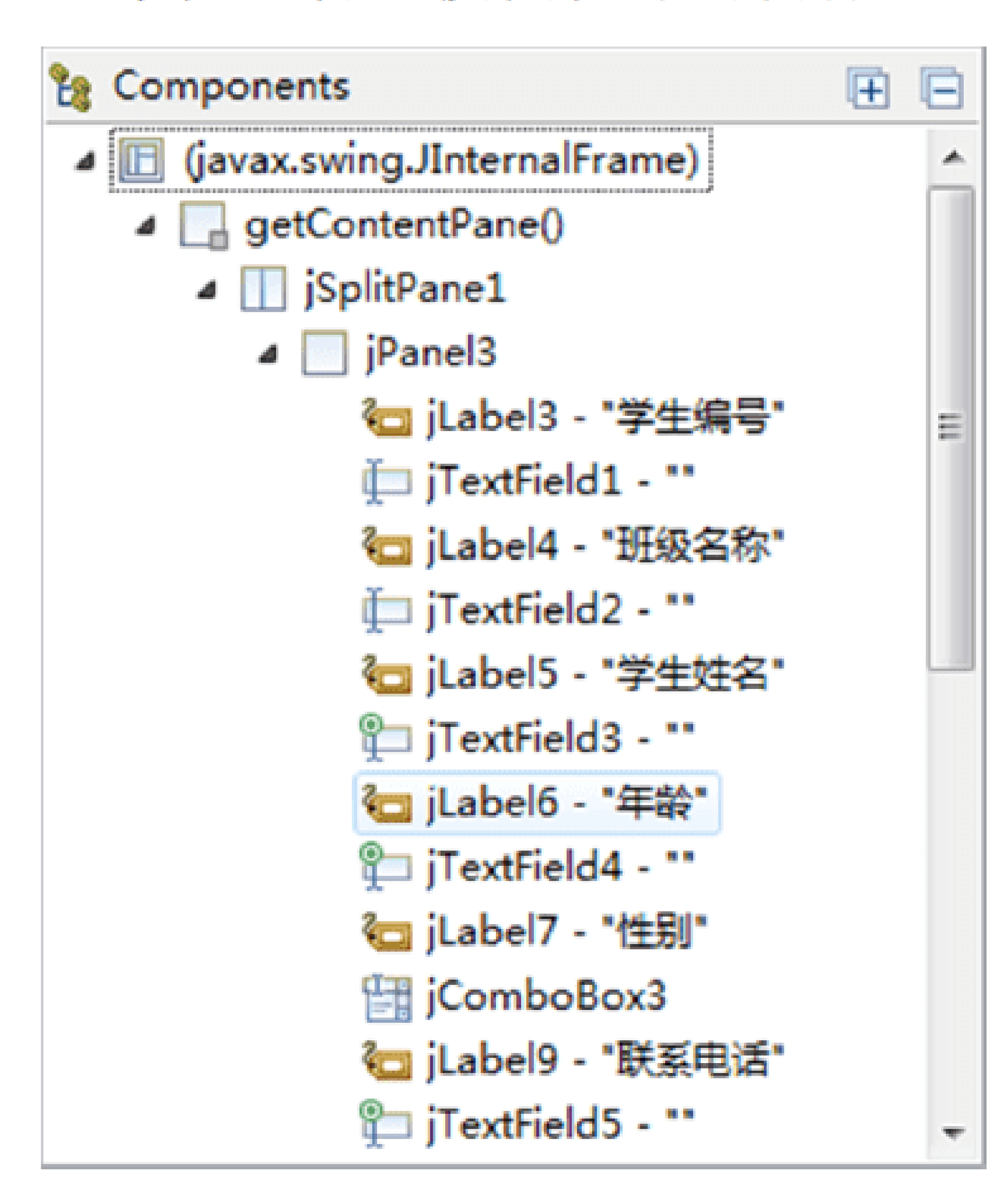


图 21.17　JF_view_student 类中控件的名称（上半部分）

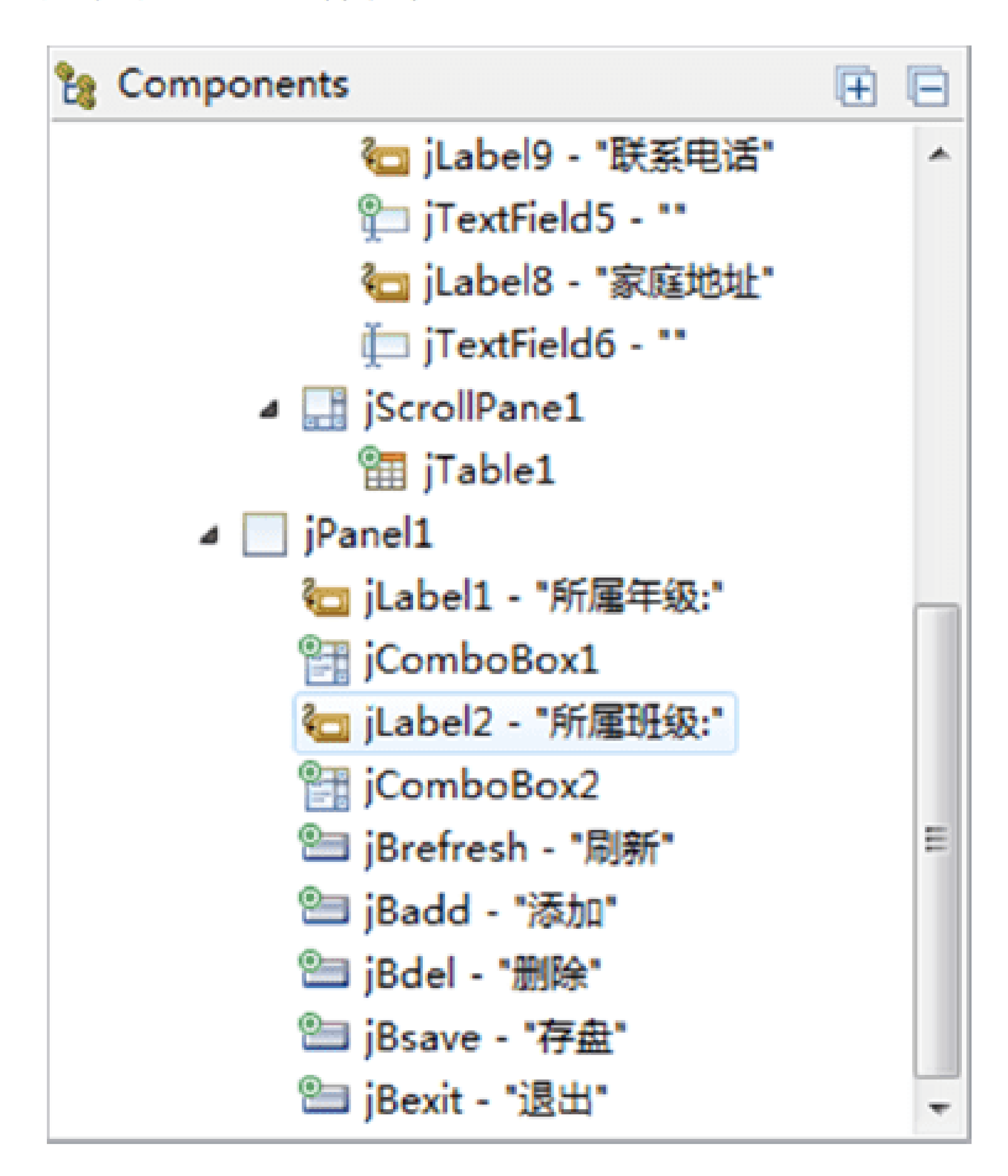


图 21.18 JF_view_student 类中控件的名称（下半部分）

2．代码设计

（1）用户进入该模块后，程序首先从数据表中检索出学生的基本信息，如果检索到学生的基本信息，那么用户在单击某一条数据之后可以对该数据进行修改、删除等操作，公共类 JdbcAdapter 是对学生信息表 tb_studentinfo 进行相应操作。下面请读者来看一下检索数据的功能，单击 JF_view_student 类的 Source 代码编辑窗口，首先导入 util 公共包下的相应类文件，定义两个 String 类型的数组变量

gradeID，classID 其初始值为 null，用来存储年级编号和班级编号，然后定义一个公有方法 initialize 用来检索班级数据，其关键代码如下：

```
public void initialize() {
    String sqlStr = null;
    sqlStr = "select gradeID,gradeName from tb_gradeinfo";
    RetrieveObject retrieve = new RetrieveObject();
    java.util.Collection collection = null;
    java.util.Iterator iterator = null;
    collection = retrieve.getTableCollection(sqlStr);
    iterator = collection.iterator();
    gradeID = new String[collection.size()];
    int i = 0;
    while (iterator.hasNext()) {
        java.util.Vector vdata = (java.util.Vector) iterator.next();
        gradeID[i] = String.valueOf(vdata.get(0));
        jComboBox1.addItem(vdata.get(1));
        i++;
    }
}
```

（2）用户选择年级列表框（jComboBox1）数据后，系统会自动检索出年级下面的班级数据，并放入到班级列表框（jComboBox2）中，在公共方法 jComboBox1_itemStateChanged 中，定义一个 String 类型变量 sqlStr，用来存储 SQL 查询语句，执行公共类 RetrieveObject 的方法 getTableCollection，其参数为 sqlStr，将返回值放入集合变量 collection 中，然后将集合中的数据存放到班级列表框控件中，其关键代码如下：

```
public void jComboBox1_itemStateChanged(ItemEvent e) {
    jComboBox2.removeAllItems();
    int Index = jComboBox1.getSelectedIndex();
    String sqlStr = null;
    sqlStr = "select classID,className from tb_classinfo where gradeID = '" + gradeID[Index] + "'";
    RetrieveObject retrieve = new RetrieveObject();
    java.util.Collection collection = null;
    java.util.Iterator iterator = null;
    collection = retrieve.getTableCollection(sqlStr);
    iterator = collection.iterator();
    classID = new String[collection.size()];
    int i = 0;
    while (iterator.hasNext()) {
        java.util.Vector vdata = (java.util.Vector) iterator.next();
        classID[i] = String.valueOf(vdata.get(0));
        jComboBox2.addItem(vdata.get(1));
        i++;
    }
}
```

（3）用户选择班级列表框（jComboBox2）数据后，系统自动检索出该班级下的所有学生数据，方法 jComboBox2_itemStateChanged 的关键代码如下：

```
public void jComboBox2_itemStateChanged(ItemEvent e) {
    if (jComboBox2.getSelectedIndex() < 0)
        return;
    String cid = classID[jComboBox2.getSelectedIndex()];
    DefaultTableModel tablemodel = null;
    String[] name = { "学生编号", "班级编号", "学生姓名", "性别", "年龄", "家庭住址", "联系电话" };
    String sqlStr = "select * from tb_studentinfo where classid = '" + cid + "'";
    appstu.util.RetrieveObject bdt = new appstu.util.RetrieveObject();
    tablemodel = bdt.getTableModel(name, sqlStr);
    jTable1.setModel(tablemodel);
    jTable1.setRowHeight(24);
}
```

（4）用户单击表格中的某条数据后，系统会将学生的信息读取到面板 jPanel1 的控件上来，以供用户进行操作，其关键代码如下：

```
public void jTable1_mouseClicked(MouseEvent e) {
    String id = null;
    String sqlStr = null;
    int selectrow = 0;
    selectrow = jTable1.getSelectedRow();
    if (selectrow < 0)
        return;
    id = jTable1.getValueAt(selectrow, 0).toString();
    sqlStr = "select * from tb_studentinfo where stuid = '" + id + "'";
    Vector vdata = null;
    RetrieveObject retrive = new RetrieveObject();
    vdata = retrive.getObjectRow(sqlStr);
    String gradeid = null, classid = null;
    String gradename = null, classname = null;
    Vector vname = null;
    classid = vdata.get(1).toString();
    gradeid = classid.substring(0, 2);
    vname = retrive.getObjectRow("select className from tb_classinfo where classID = '" + classid + "'");
    classname = String.valueOf(vname.get(0));
    vname = retrive.getObjectRow("select gradeName from tb_gradeinfo where gradeID = '" + gradeid + "'");
    gradename = String.valueOf(vname.get(0));
    jTextField1.setText(vdata.get(0).toString());
    jTextField2.setText(gradename + classname);
    jTextField3.setText(vdata.get(2).toString());
    jTextField4.setText(vdata.get(4).toString());
    jTextField5.setText(vdata.get(6).toString());
    jTextField6.setText(vdata.get(5).toString());
```

```
        jComboBox3.removeAllItems();
        jComboBox3.addItem(vdata.get(3).toString());
}
```

（5）单击“添加”按钮，进行录入操作，这里我们主要看一下最大流水号的生成，其中公共方法 jBadd_actionPerformed 的关键代码如下：

```
public void jBadd_actionPerformed(ActionEvent e) {
    String classid = null;
    int index = jComboBox2.getSelectedIndex();
    if (index < 0) {
        JOptionPane.showMessageDialog(null, "班级名称为空,请重新选择班级", "系统提示",
                                      JOptionPane.ERROR_MESSAGE);
        return;
    }
    classid = classID[index];
    String sqlMax = "select max(stuid) from tb_studentinfo where classID = '" + classid + "'";
    ProduceMaxBh pm = new appstu.util.ProduceMaxBh();
    String stuid = null;
    stuid = pm.getMaxBh(sqlMax, classid);
    jTextField1.setText(stuid);
    jTextField2.setText(jComboBox2.getSelectedItem().toString());
    jTextField3.setText("");
    jTextField4.setText("");
    jTextField5.setText("");
    jTextField6.setText("");
    jComboBox3.removeAllItems();
    jComboBox3.addItem("男");
    jComboBox3.addItem("女");
    jTextField3.requestFocus();
}
```

（6）单击“删除”按钮，删除学生信息，其中公共方法 jBdel_actionPerformed 的关键代码如下：

```
public void jBdel_actionPerformed(ActionEvent e) {
    if (jTextField1.getText().trim().length() <= 0)
        return;
    int result = JOptionPane.showOptionDialog(null, "是否删除学生的基本信息数据?", "系统提示",
       JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, new String[] {"是", "否" }, "否");
    if (result == JOptionPane.NO_OPTION)
        return;
    String sqlDel = "delete tb_studentinfo where stuid = '" + jTextField1.getText().trim() + "'";
    JdbcAdapter jdbcAdapter = new JdbcAdapter();
    if (jdbcAdapter.DeleteObject(sqlDel)) {
        jTextField1.setText("");
        jTextField2.setText("");
```

```
            jTextField3.setText("");
            jTextField4.setText("");
            jTextField5.setText("");
            jTextField6.setText("");
            jComboBox1.removeAllItems();
            jComboBox3.removeAllItems();
            ActionEvent event = new ActionEvent(jBrefresh, 0, null);
            jBrefresh_actionPerformed(event);
        }
}
```

（7）单击“存盘”按钮，对数据进行存盘操作，其中公共方法 jBsave_actionPerformed 的关键代码如下：

```
public void jBsave_actionPerformed(ActionEvent e) {
        int result = JOptionPane.showOptionDialog(null, "是否存盘学生基本数据信息?", "系统提示",
            JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, new String[] { "是", "否" }, "否");
        if (result == JOptionPane.NO_OPTION)
            return;
        appstu.model.Obj_student object = new appstu.model.Obj_student();
        String classid = classID[Integer.parseInt(String.valueOf(jComboBox2.getSelectedIndex()))];
        object.setStuid(jTextField1.getText().trim());
        object.setClassID(classid);
        object.setStuname(jTextField3.getText().trim());
        int age = 0;
        try {
            age = Integer.parseInt(jTextField4.getText().trim());
        } catch (java.lang.NumberFormatException formate) {
            JOptionPane.showMessageDialog(null, "数据录入有误，错误信息:\n" + formate.getMessage(),
                                                    "系统提示", JOptionPane.ERROR_MESSAGE);
            jTextField4.requestFocus();
            return;
        }
        object.setAge(age);
        object.setSex(String.valueOf(jComboBox3.getSelectedItem()));
        object.setPhone(jTextField5.getText().trim());
        object.setAddress(jTextField6.getText().trim());
        appstu.util.JdbcAdapter adapter = new appstu.util.JdbcAdapter();
        if (adapter.InsertOrUpdateObject(object)) {
            ActionEvent event = new ActionEvent(jBrefresh, 0, null);
            jBrefresh_actionPerformed(event);
        }
}
```

21.10　学生考试成绩信息管理模块设计

21.10.1　学生考试成绩信息管理模块概述

学生考试成绩信息管理模块主要是对学生成绩信息进行管理，包括修改、添加、删除、存盘等。单击菜单“基本信息”/“考试成绩”选项，进入该模块，运行结果如图 21.19 所示。

图 21.19　学生考试成绩信息管理窗体

21.10.2　学生考试成绩管理模块技术分析

学生考试成绩信息管理模块使用的主要技术是 Vector 类的应用。Vector 类可以实现长度可变的对象数组。与数组一样，它包含可以使用整数索引进行访问的控件。但是，Vector 的大小可以根据需要增大或缩小，以适应创建 Vector 后进行添加或移除项的操作。

每个 Vector 对象会试图通过维护 capacity 和 capacityIncrement 来优化存储管理。capacity 始终至少与 Vector 的大小相等；这个值通常比后者大些，因为随着将控件添加到 Vector 中，其存储将按 capacityIncrement 的大小增加存储块。应用程序可以在插入大量控件前增加 Vector 的容量；这样就减少了增加的重分配的量。



21.10.3　学生考试成绩信息管理模块实现过程

1．界面设计

学生考试成绩信息管理模块设计的窗体 UI 结构图如图 21.20 所示。



图 21.20　JF_view_gradesub 类中控件的名称

2．代码设计

（1）该模块初始化时，首先获取所有的考试种类和班级，显示在 JComboBox 下拉列表中，然后获取当前的日期，对开始日期文本框的值进行初始化，代码如下：

```
public void initialize() {
    RetrieveObject retrieve = new RetrieveObject();
    java.util.Vector vdata = new java.util.Vector();
    String sqlStr = null;
    java.util.Collection collection = null;
    java.util.Iterator iterator = null;
    sqlStr = "SELECT * FROM tb_examkinds";
    collection = retrieve.getTableCollection(sqlStr);
    iterator = collection.iterator();
    examkindid = new String[collection.size()];
    examkindname = new String[collection.size()];
    int i = 0;
    while (iterator.hasNext()) {
        vdata = (java.util.Vector) iterator.next();
        examkindid[i] = String.valueOf(vdata.get(0));
        examkindname[i] = String.valueOf(vdata.get(1));
        jComboBox1.addItem(vdata.get(1));
```

```
            i++;
        }
        sqlStr = "select * from tb_classinfo";
        collection = retrieve.getTableCollection(sqlStr);
        iterator = collection.iterator();
        classid = new String[collection.size()];
        i = 0;
        while (iterator.hasNext()) {
            vdata = (java.util.Vector) iterator.next();
            classid[i] = String.valueOf(vdata.get(0));
            jComboBox2.addItem(vdata.get(2));
            i++;
        }
        sqlStr = "select * from tb_subject";
        collection = retrieve.getTableCollection(sqlStr);
        iterator = collection.iterator();
        subjectcode = new String[collection.size()];
        subjectname = new String[collection.size()];
        i = 0;
        while (iterator.hasNext()) {
            vdata = (java.util.Vector) iterator.next();
            subjectcode[i] = String.valueOf(vdata.get(0));
            subjectname[i] = String.valueOf(vdata.get(1));

            i++;
        }
        long nCurrentTime = System.currentTimeMillis();
        java.util.Calendar calendar = java.util.Calendar.getInstance(new Locale("CN"));
        calendar.setTimeInMillis(nCurrentTime);
        int year = calendar.get(Calendar.YEAR);
        int month = calendar.get(Calendar.MONTH) + 1;
        int day = calendar.get(Calendar.DAY_OF_MONTH);
        String mm, dd;
        if (month < 10) {
            mm = "0" + String.valueOf(month);
        } else {
            mm = String.valueOf(month);
        }
        if (day < 10) {
            dd = "0" + String.valueOf(day);
        } else {
            dd = String.valueOf(day);
        }
        java.sql.Date date = java.sql.Date.valueOf(year + "-" + mm + "-" + dd);
        jTextField1.setText(String.valueOf(date));
    }
```

（2）单击学生信息表格中的某个学生的信息，如果该学生已经录入了考试成绩，检索出成绩数据信息，在公共方法 jTable1_mouseClicked 中定义一个 String 类型的局部变量 sqlStr，用来存储 SQL 的查询语句，然后调用公共类 RetrieveObject 的公共方法 getTableCollection，其参数为 sqlStr，返回值为集合 Collection，然后将集合中数据存放到表格控件中。公共方法 jTable1_mouseClicked 的关键代码如下：

```
public void jTable1_mouseClicked(MouseEvent e) {
    int currow = jTable1.getSelectedRow();
    if (currow >= 0) {
        DefaultTableModel tablemodel = null;
        String[] name = { "学生编号", "学生姓名", "考试类别", "考试科目", "考试成绩", "考试时间" };
        tablemodel = new DefaultTableModel(name, 0);
        String sqlStr = null;
        Collection collection = null;
        Object[] object = null;
        sqlStr = "SELECT * FROM tb_gradeinfo_sub where stuid = '" + jTable1.getValueAt(currow, 0)
                + "' and kindID = '"+ examkindid[jComboBox1.getSelectedIndex()] + "'";
        RetrieveObject retrieve = new RetrieveObject();
        collection = retrieve.getTableCollection(sqlStr);
        object = collection.toArray();
        int findindex = 0;
        for (int i = 0; i < object.length; i++) {
            Vector vrow = new Vector();
            Vector vdata = (Vector) object[i];
            String sujcode = String.valueOf(vdata.get(3));
            for (int aa = 0; aa < this.subjectcode.length; aa++) {
                if (sujcode.equals(subjectcode[aa])) {
                    findindex = aa;
                    System.out.println("findindex = " + findindex);
                }
            }
            if (i == 0) {
                vrow.addElement(vdata.get(0));
                vrow.addElement(vdata.get(1));
                vrow.addElement(examkindname[Integer.parseInt(String.valueOf(vdata.get(2))) - 1]);
                vrow.addElement(subjectname[findindex]);
                vrow.addElement(vdata.get(4));
                String ksrq = String.valueOf(vdata.get(5));
                ksrq = ksrq.substring(0, 10);
                System.out.println(ksrq);
                vrow.addElement(ksrq);
            } else {
                vrow.addElement("");
                vrow.addElement("");
                vrow.addElement("");
                vrow.addElement(subjectname[findindex]);
```

```
                vrow.addElement(vdata.get(4));
                String ksrq = String.valueOf(vdata.get(5));
                ksrq = ksrq.substring(0, 10);
                System.out.println(ksrq);
                vrow.addElement(ksrq);
            }
            tablemodel.addRow(vrow);
        }
        this.jTable2.setModel(tablemodel);
        this.jTable2.setRowHeight(22);
    }
}
```

（3）单击学生信息表格中的某个学生信息，如果没有检索到学生的成绩数据，单击“添加”按钮，进行成绩数据的添加，在公共方法 jBadd_actionPerformed 中定义一个表格模型 DefaultTableModel 变量 tablemodel，用来生成数据表格。定义一个 String 类型的局部变量 sqlStr，用来存放查询语句，调用公共类 RetrieveObject 的 getObjectRow 方法，其参数为 sqlStr，用返回类型 vector 生成科目名称，然后为 tablemodel 填充数据，关键代码如下：

```
public void jBadd_actionPerformed(ActionEvent e) {
    int currow;
    currow = jTable1.getSelectedRow();
    if (currow >= 0) {
        DefaultTableModel tablemodel = null;
        String[] name = { "学生编号", "学生姓名", "考试类别", "考试科目", "考试成绩", "考试时间" };
        tablemodel = new DefaultTableModel(name, 0);
        String sqlStr = null;
        Collection collection = null;
        Object[] object = null;
        Iterator iterator = null;
        sqlStr = "SELECT subject FROM tb_subject";                    // 定义查询参数
        RetrieveObject retrieve = new RetrieveObject();               // 定义公共类对象
        Vector vdata = null;
        vdata = retrieve.getObjectRow(sqlStr);
        for (int i = 0; i < vdata.size(); i++) {
            Vector vrow = new Vector();
            if (i == 0) {
                vrow.addElement(jTable1.getValueAt(currow, 0));
                vrow.addElement(jTable1.getValueAt(currow, 2));
                vrow.addElement(jComboBox1.getSelectedItem());
                vrow.addElement(vdata.get(i));
                vrow.addElement("");
                vrow.addElement(jTextField1.getText().trim());
            } else {
                vrow.addElement("");
```

```
                vrow.addElement("");
                vrow.addElement("");
                vrow.addElement(vdata.get(i));
                vrow.addElement("");
                vrow.addElement(jTextField1.getText().trim());
            }
            tablemodel.addRow(vrow);
            this.jTable2.setModel(tablemodel);
            this.jTable2.setRowHeight(23);
        }
    }
}
```

（4）输入完学生成绩数据后，单击“存盘”按钮，进行数据存盘。在公共方法 jBsave_actionPerformed 中定义一个类型为对象 Obj_gradeinfo_sub 数组变量 object，通过循环语句为 object 变量中的对象赋值，然后调用公共类 jdbcAdapter 中的 InsertOrUpdate_Obj_ gradeinfo_sub 方法，其参数为 object，执行存盘操作，关键代码如下：

```
public void jBsave_actionPerformed(ActionEvent e) {
    int result = JOptionPane.showOptionDialog(null, "是否存盘学生考试成绩数据?", "系统提示",
       JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, new String[] {"是", "否" }, "否");
    if (result == JOptionPane.NO_OPTION)
        return;
    int rcount;
    rcount = jTable2.getRowCount();
    if (rcount > 0) {
        appstu.util.JdbcAdapter jdbcAdapter = new appstu.util.JdbcAdapter();
        Obj_gradeinfo_sub[] object = new Obj_gradeinfo_sub[rcount];
        for (int i = 0; i < rcount; i++) {
            object[i] = new Obj_gradeinfo_sub();
            object[i].setStuid(String.valueOf(jTable2.getValueAt(0, 0)));
            object[i].setKindID(examkindid[jComboBox1.getSelectedIndex()]);
            object[i].setCode(subjectcode[i]);
            object[i].setSutname(String.valueOf(jTable2.getValueAt(i, 1)));
            float grade;
            grade = Float.parseFloat(String.valueOf(jTable2.getValueAt(i, 4)));
            object[i].setGrade(grade);
            java.sql.Date rq = null;
            try {
                String strrq = String.valueOf(jTable2.getValueAt(i, 5));
                rq = java.sql.Date.valueOf(strrq);
            } catch (Exception dt) {
                JOptionPane.showMessageDialog(null, "第【" + i + "】行输入的数据格式有误,请重新录入!!\n"
                        + dt.getMessage(), "系统提示", JOptionPane.ERROR_MESSAGE);
                return;
```

```
                }
                object[i].setExamdate(rq);
            }
            jdbcAdapter.InsertOrUpdate_Obj_gradeinfo_sub(object);          // 执行公共类中的数据存盘操作
        }
}
```

21.11　基本信息数据查询模块设计

21.11.1　基本信息数据查询模块概述

基本信息数据查询包括对学生信息查询和教师信息查询两部分，单击菜单“系统查询”/“基本信息”选项，进入该模块，其运行结果如图 21.21 所示。



图 21.21　基本信息数据查询窗体

21.11.2　基本信息数据查询模块技术分析

在标准 SQL 中，定义了模糊查询。它是使用 LIKE 关键字完成的。模糊查询的重点在于两个符号的使用：%和_。%表示任意多个字符，_表示任意一个字符。例如在姓名列中查询条件是“王%”，那么可以找到所有王姓同学；如果查询条件是“王_”，那么可以找到名的长度为 1 的王姓同学。

21.11.3　基本信息数据查询模块实现过程

1．界面设计

基本信息数据查询模块设计的窗体 UI 结构图如图 21.22 所示。



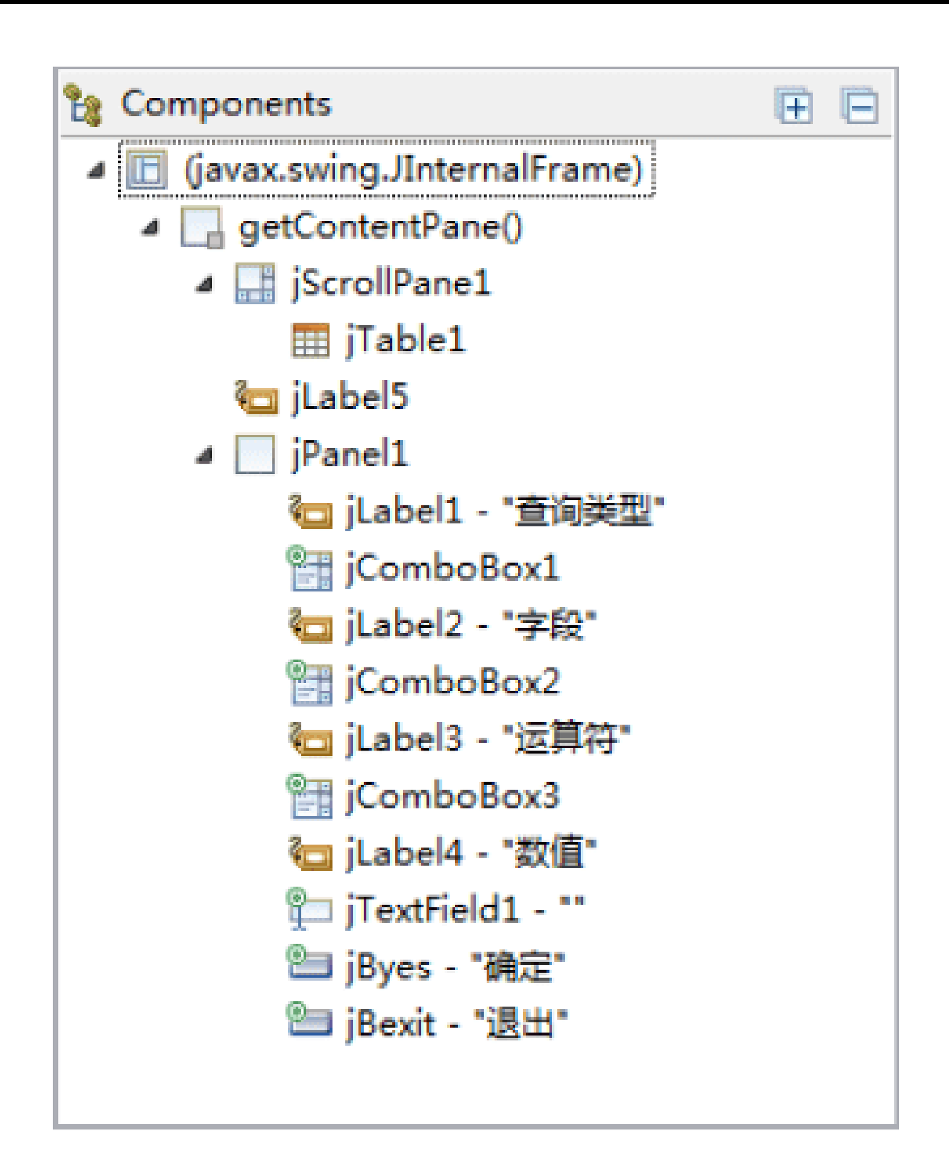


图 21.22　JF_view_query_jbqk 类中控件的名称

2．代码设计

（1）用户首先选择查询类型，也就是选择查询什么信息，然后根据系统提供的查询参数进行条件选择，输入查询数值之后，单击“确定”按钮，进行满足条件的数据查询。单击 Source 页打开文件源代码，导入程序所需要的类包，定义不同的 String 类型变量，定义一个私有方法 initsize 用来初始化列表框中的数据，以供用户选择条件参数，关键代码如下：

```
public class JF_view_query_jbqk extends JInternalFrame {
    String tabname = null;
    String zdname = null;
    String ysfname = null;
    String[] jTname = null;
    private void initsize() {
        jComboBox1.addItem("学生信息");
        jComboBox1.addItem("教师信息");
        jComboBox3.addItem("like");
        jComboBox3.addItem(">");
        jComboBox3.addItem("=");
        jComboBox3.addItem("<");
        jComboBox3.addItem(">=");
        jComboBox3.addItem("<=");
    }
}
```

（2）用户选择不同的查询类型系统时为查询字段列表框进行字段赋值，在公共方法 jComboBox1_itemStateChanged 中实现这个功能，关键代码如下：

```
public void jComboBox1_itemStateChanged(ItemEvent itemEvent) {
    if (jComboBox1.getSelectedIndex() == 0) {
        this.tabname = "SELECT s.stuid, c.className, s.stuname, s.sex, s.age, s.addr, s.phone FROM
                                    tb_studentinfo s ,tb_classinfo c where s.classID = c.classID";
        String[] name = { "学生编号", "班级名称", "学生姓名", "性别", "年龄", "家庭住址", "联系电话" };
        jTname = name;
```

```
            jComboBox2.removeAllItems();
            jComboBox2.addItem("学生编号");
            jComboBox2.addItem("班级编号");
      }
      if (jComboBox1.getSelectedIndex() == 1) {
            this.tabname = "SELECT t.teaid, c.className, t.teaname, t.sex, t.knowledge, t.knowlevel FROM "+
                                    " tb_teacher t INNER JOIN tb_classinfo c ON c .classID = t.classID";
            String[] name = { "教师编号", "班级名称", "教师姓名", "性别", "教师职称", "教师等级" };
            jTname = name;
            jComboBox2.removeAllItems();
            jComboBox2.addItem("教师编号");
            jComboBox2.addItem("班级编号");
      }
}
```

（3）用户选择不同的查询字段之后，程序为实例变量 zdname 进行赋值，其公共方法 jComboBox2_itemStateChanged 的关键代码如下：

```
public void jComboBox2_itemStateChanged(ItemEvent itemEvent) {
      if (jComboBox1.getSelectedIndex() == 0) {
            if (jComboBox2.getSelectedIndex() == 0)
                  this.zdname = "s.stuid";
            if (jComboBox2.getSelectedIndex() == 1)
                  this.zdname = "s.classID";
      }
      if (jComboBox1.getSelectedIndex() == 1) {
            if (jComboBox2.getSelectedIndex() == 0)
                  this.zdname = "t.teaid";
            if (jComboBox2.getSelectedIndex() == 1)
                  this.zdname = "t.classID";
      }
      System.out.println("zdname = " + zdname);
}
```

（4）同样，当用户选择不同的运算符之后程序为实例变量 ysfname 进行赋值，其公共方法 jComboBox3_itemStateChanged 的关键代码如下：

```
public void jComboBox3_itemStateChanged(ItemEvent itemEvent) {
      this.ysfname = String.valueOf(jComboBox3.getSelectedItem());
}
```

（5）用户输入检索数值之后，单击“确定”按钮，进行条件查询操作。在公共方法 jByes_actionPerformed 中，定义两个 String 类型局部变量 sqlSelect 与 whereSql，用来生成查询条件语句。通过公共类 RetrieveObject 的 getTableModel 方法，进行查询操作，其参数为 sqlSelect 和 whereSql，其详细代码如下：

```
public void jByes_actionPerformed(ActionEvent e) {
      String sqlSelect = null, whereSql = null;
      String valueStr = jTextField1.getText().trim();
```



```
        sqlSelect = this.tabname;
        if (ysfname == "like") {
            whereSql = " and " + this.zdname + " " + this.ysfname + " '%" + valueStr + "%'";
        } else {
            whereSql = " and " + this.zdname + " " + this.ysfname + " '" + valueStr + "'";
        }
        appstu.util.RetrieveObject retrieve = new appstu.util.RetrieveObject();
        javax.swing.table.DefaultTableModel defaultmodel = null;
        defaultmodel = retrieve.getTableModel(jTname, sqlSelect + whereSql);
        jTable1.setModel(defaultmodel);
        if (jTable1.getRowCount() <= 0) {
            JOptionPane.showMessageDialog(null, "没有找到满足条件的数据!!!", "系统提示",
                                                            JOptionPane.INFORMATION_MESSAGE);
        }
        jTable1.setRowHeight(24);
        jLabel5.setText("共有数据【" + String.valueOf(jTable1.getRowCount()) + "】条");
}
```

21.12　考试成绩班级明细数据查询模块设计

21.12.1　考试成绩班级明细数据查询模块概述

考试成绩班级明细数据查询模块用来查询不同班级的学生考试明细信息，其运行结果如图 21.23 所示。



学生编号	学生姓名	数学	外语	语文	政治	历史	体育
010202	刘*笑	85.6	78.9	95.8	99.2	76.5	88.4
010201	刑*庆	85.5	92.5	90.6	82.5	70.5	66.5

图 21.23　考试成绩班级明细数据查询窗体

21.12.2　考试成绩班级明显数据查询模块技术分析

在 Java 中，如果开发桌面应用程序，通常使用 Swing。Swing 中的控件大都有其默认的设置，例如 JTable 控件在创建完成后，表格内容的行高就有了一个固定值。如果修改了表格文字的字体，则可能影响正常显示。此时可以考虑使用 JTable 控件中提供的 setRowHeight 方法重新设置行高。该方法的声明如下：

```
public void setRowHeight(int rowHeight)
```

rowHeight：新的行高。

21.12.3　考试成绩班级明细数据查询模块实现过程

1．界面设计

考试成绩班级明细数据查询模块设计的窗体 UI 结构图如图 21.24 所示。

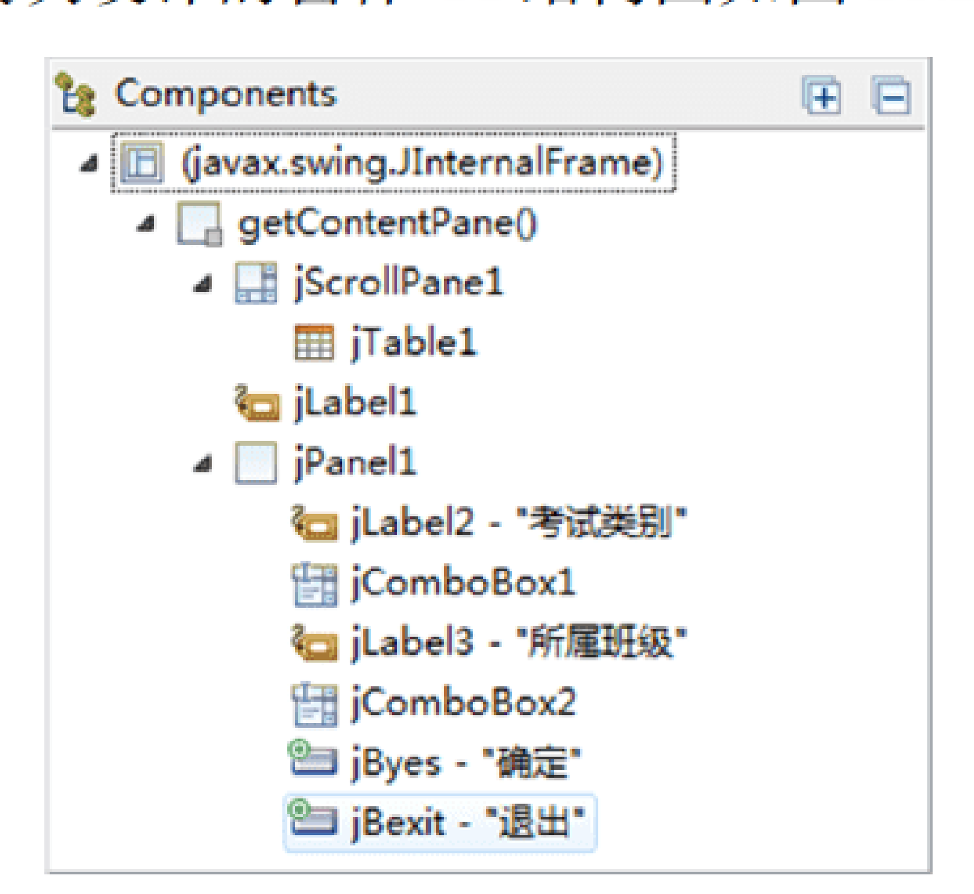


图 21.24　JF_view_query_grade_mx 类中控件的名称

2．代码设计

（1）定义一个私有方法 initsize，用来初始化列表框中的数据，供用户选择条件参数，关键代码如下：

```
public class JF_view_query_grade_mx extends JInternalFrame {
    String classid[] = null;
    String classname[] = null;
    String examkindid[] = null;
    String examkindname[] = null;
    public void initialize() {
        RetrieveObject retrieve = new RetrieveObject();
        java.util.Vector vdata = new java.util.Vector();
        String sqlStr = null;
        java.util.Collection collection = null;
        java.util.Iterator iterator = null;
        sqlStr = "SELECT * FROM tb_examkinds";
```



```
            collection = retrieve.getTableCollection(sqlStr);
            iterator = collection.iterator();
            examkindid = new String[collection.size()];
            examkindname = new String[collection.size()];
            int i = 0;
            while (iterator.hasNext()) {
                vdata = (java.util.Vector) iterator.next();
                examkindid[i] = String.valueOf(vdata.get(0));
                examkindname[i] = String.valueOf(vdata.get(1));
                jComboBox1.addItem(vdata.get(1));
                i++;
            }
            sqlStr = "select * from tb_classinfo";
            collection = retrieve.getTableCollection(sqlStr);
            iterator = collection.iterator();
            classid = new String[collection.size()];
            classname = new String[collection.size()];
            i = 0;
            while (iterator.hasNext()) {
                vdata = (java.util.Vector) iterator.next();
                classid[i] = String.valueOf(vdata.get(0));
                classname[i] = String.valueOf(vdata.get(2));
                jComboBox2.addItem(vdata.get(2));
                i++;
            }
        }
        // 省略部分代码
}
```

（2）用户选择“考试类别”和“所属班级”后，单击“确定”按钮，进行成绩明细数据查询。在公共方法 jByes_actionPerformed 中，定义一个 String 类型的局部变量 sqlSubject，用来存储考试科目的查询语句；定义一个 String 类型数组变量 tbname，用来为表格模型设置列的名字。定义公共类 RetrieveObject 的变量 retrieve，然后执行 retrieve 的方法 getTableCollection，其参数为 sqlSubject。当结果集中存在数据的时候，定义一个 String 变量 sqlStr，用来生成查询成绩的语句，通过一个循环语句为 sqlStr 赋值，再定义一个公共类 RetrieveObject 类型的变量 bdt，执行 bdt 的 getTableModel 方法，其参数为 tbname 和 sqlStr 变量。公共方法 jByes_actionPerformed 的关键代码如下：

```
public void jByes_actionPerformed(ActionEvent e) {
    String sqlSubject = null;
    java.util.Collection collection = null;
    Object[] object = null;
    java.util.Iterator iterator = null;
    sqlSubject = "SELECT * FROM tb_subject";
    RetrieveObject retrieve = new RetrieveObject();
    collection = retrieve.getTableCollection(sqlSubject);
    object = collection.toArray();
    String strCode[] = new String[object.length];                  // 定义数组存放考试科目代码
    String strSubject[] = new String[object.length];               // 定义数组存放考试科目名称
```

```
    String[] tbname = new String[object.length + 2];                     // 定义数组存放表格控件的列名
    tbname[0] = "学生编号";
    tbname[1] = "学生姓名";
    String sqlStr = "SELECT stuid, stuname, ";
    for (int i = 0; i < object.length; i++) {
        String code = null, subject = null;
        java.util.Vector vdata = null;
        vdata = (java.util.Vector) object[i];
        code = String.valueOf(vdata.get(0));
        subject = String.valueOf(vdata.get(1));
        tbname[i + 2] = subject;
        if ((i + 1) == object.length) {
            sqlStr = sqlStr + " SUM(CASE code WHEN '" + code + "'
                    THEN grade ELSE 0 END) AS '" + subject + "'";
        } else {
            sqlStr = sqlStr + " SUM(CASE code WHEN '" + code + "'
                    THEN grade ELSE 0 END) AS '" + subject + "',";
        }
    }
    String whereStr = " where kind";
    // 为变量 whereStr 进行赋值操作生成查询的 SQL 语句
    whereStr = " where kindID = '" + this.examkindid[jComboBox1.getSelectedIndex()] + "' and substring
(stuid,1,4) = '"
            + this.classid[jComboBox2.getSelectedIndex()] + "' ";
    // 为变量 sqlStr 进行赋值操作生成查询的 SQL 语句
    sqlStr = sqlStr + " FROM tb_gradeinfo_sub " + whereStr + " GROUP BY stuid,stuname ";
    DefaultTableModel tablemodel = null;
    appstu.util.RetrieveObject bdt = new appstu.util.RetrieveObject();
    tablemodel = bdt.getTableModel(tbname, sqlStr);     // 通过对象 bdt 的 getTableModel 方法为表格赋值
    jTable1.setModel(tablemodel);
    if (jTable1.getRowCount() <= 0) {
        JOptionPane.showMessageDialog(null, "没有找到满足条件的数据!!!", "系统提示",
                                      JOptionPane.INFORMATION_MESSAGE);
    }
    jTable1.setRowHeight(24);
    jLabel1.setText("共有数据【" + String.valueOf(jTable1.getRowCount()) + "】条");
}
```

21.13 小　结

本章从软件工程的角度，讲述开发软件的常规步骤。在学生成绩管理系统的开发过程中，读者应该掌握使用 Java 的 Swing 技术进行开发的一般过程。此外，对于 JDBC 等常用技术也应该有更加深入的了解。

第 22 章　图书商城（Java Web+ SQL Server 2014 实现）

在“倡导全民阅读，建设书香社会”的大背景下，每个人都会或多或少地购买一些图书。而网购图书以其方便、快捷等优点备受大家欢迎。因此，本章将使用 JSP 技术实现一个图书商城。通过本章的学习，可以掌握以下要点：

- ☑ JavaScript 的基本应用
- ☑ JSP 文件的编写
- ☑ Servlet 的配置
- ☑ JavaBean 的编写方法
- ☑ JDBC 数据库的连接方法

22.1　开 发 背 景

纵观当下，网络已经成为现代人生活中的一部分，网络购物已深入人心，越来越多的人喜欢在网上交易。对于图书销售行业也不例外，它已由传统的书店，渐渐向网上书店转化。与传统的书店相比，网上书店可以节省商场租金、书本上架、书本翻阅损耗和员工工资等很大一笔成本。降低成本后，体现在用户身上便是低价格。这就带来更多的用户群体，从而给网上书店的发展带来了更大的优势。

22.2　系 统 分 析

22.2.1　需求分析

图书商城是基于 B/S 模式的电子商务网站，用于满足不同人群的购书需求，笔者通过对现有的商务网站的考察和研究，从经营者和消费者的角度出发，以高效管理、满足消费者需求为原则，要求本系统满足以下要求：

- ☑ 统一友好的操作界面，具有良好的用户体验；
- ☑ 图书分类详尽，可按不同类别查看图书信息；
- ☑ 最新上架图书和打折图书的展示；

☑ 会员信息的注册及验证；
☑ 用户可通过关键字搜索指定的产品信息；
☑ 用户可通过购物车一次购买多件商品；
☑ 实现收银台的功能，用户选择商品后可以在线提交订单；
☑ 提供简单的安全模型，用户必须先登录，才允许购买商品；
☑ 用户可查看自己的订单信息；
☑ 设计网站后台，管理网站的各项基本数据；
☑ 系统运行安全稳定、响应及时。

22.2.2 可行性分析

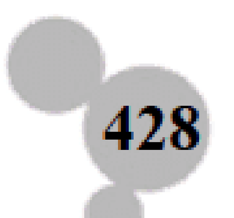
传统渠道销售图书，经常出现以下情况：
☑ 需要开设实体店铺，租金昂贵。
☑ 需要顾客主动进入书店购书。
☑ 需要店员手动记录日记账，工作量大。
☑ 店铺不仅需要开设电子支付，还要准备零钱和 POS 机。
☑ 由于商品量较大，经常出现错登记与漏登记的情况。
☑ 实体书店需要对图书进行分类摆放。
☑ 只能通过现场清点商品了解库存信息。
☑ 对库存、人员、采购等内容都是分类统计，不利于管理。

因此，从经营者的角度来看，将销售图书的渠道转移到互联网上，不仅可以节省成本，还更方便于用户查找。这样以少量的人力资源、高效的工作效率、最低的误差进行管理，将使图书销售做得更好，让顾客更信赖商家。

22.3 系 统 设 计

22.3.1 系统目标

根据电商平台要求，制定图书商城目标如下：
☑ 灵活的人机交互界面，操作简单方便，界面简洁美观。
☑ 对采购信息进行统计分析。
☑ 对超市基本档案进行管理，并提供类别统计功能。
☑ 实现各种查询，如多条件查询、模糊查询等。
☑ 提供日历功能，方便用户查询日期。
☑ 提供超市人员管理功能。
☑ 系统运行稳定、安全可靠。



22.3.2　系统功能结构

图书商城共分为两个部分，前台和后台。前台主要实现图书展示及销售；后台主要是对商城中的图书信息、会员信息，以及订单信息进行有效管理等。其详细功能结构如图 22.1 所示。

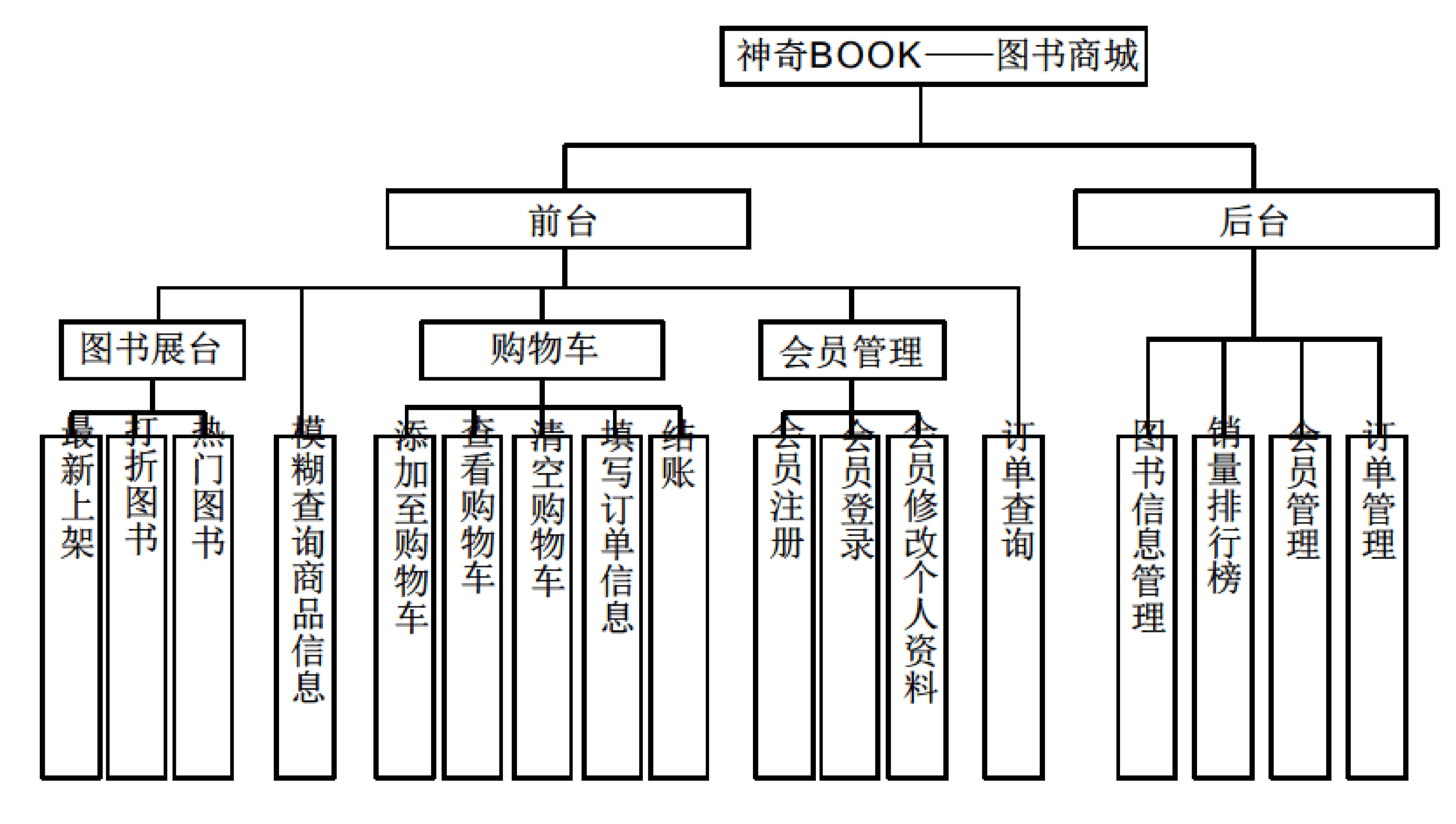


图 22.1　神奇 Book——图书商城的功能结构

22.3.3　系统流程图

在开发图书商城前，需要先了解图书商城的业务流程。根据对其他图书商城的业务分析，并结合自己的需求，设计出图 22.2 所示的图书商城的业务流程图。

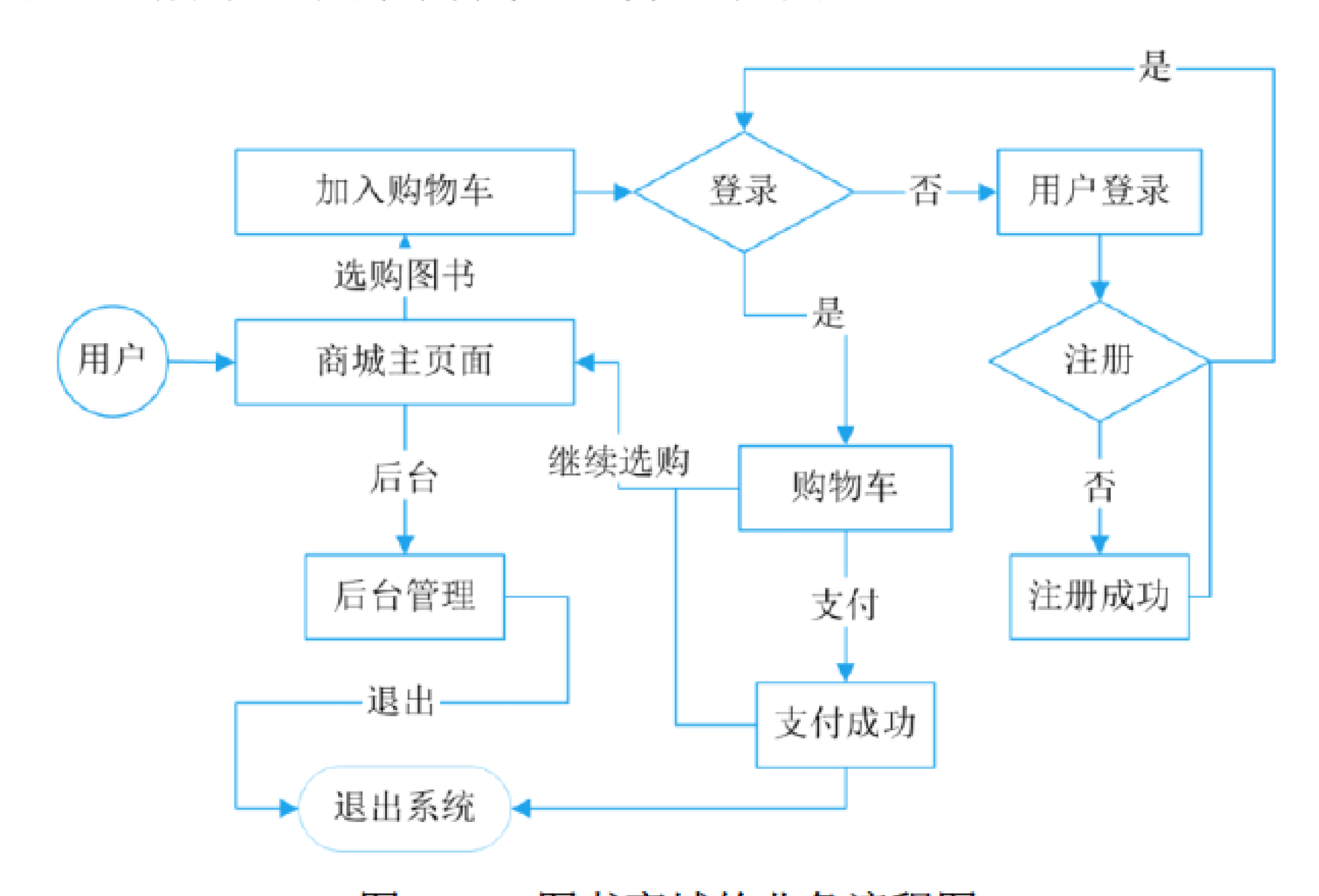


图 22.2　图书商城的业务流程图

22.3.4 系统预览

作为电商平台，图书商城提供了非常丰富的页面，根据功能模块分类如下。

图书商城的主页面如图 22.3 所示。单击具体图书链接之后可以打开图书详细信息页面，如图 22.4 所示。



图 22.3　前台首页



图 22.4　查看图书详细信息页面

如果用户想要购买图书，需要先登录图书商城，登录页面如图 22.5 所示。如果用户没有账号，需要先注册，注册页面如图 22.6 所示。



图 22.5　会员登录页面



图 22.6　会员注册页面



登录之后，可以查看自己的购物车，购物车页面如图 22.7 所示。确定完订单之后使用支付宝支付，可以看到图 22.8 所示对话框。

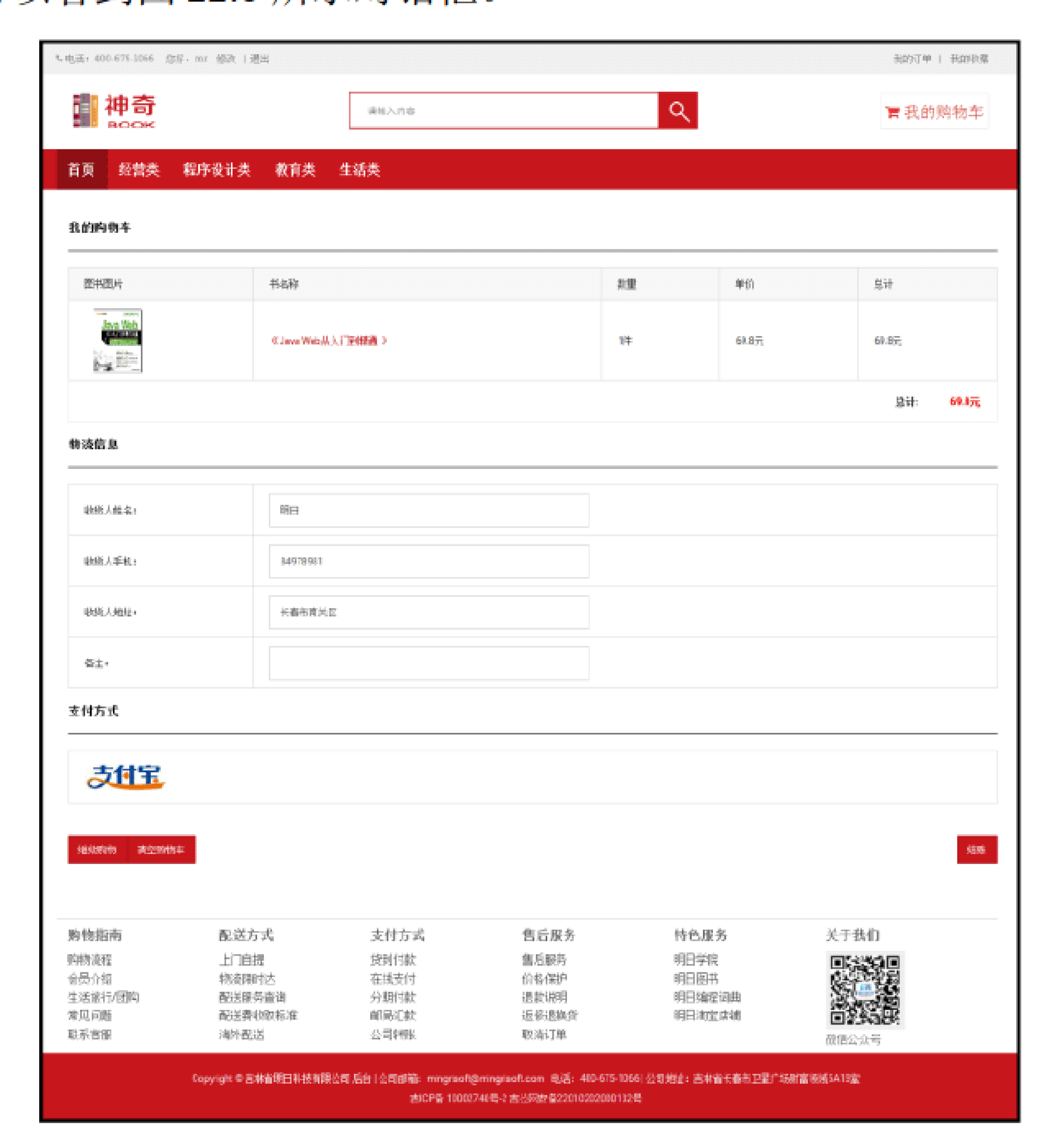



图 22.7　查看购物车页面



图 22.8　支付对话框

后台管理员可以对商城商品进行更新维护，管理员登录页面如图 22.9 所示，登录之后的首页如图 22.10 所示。



图 22.9　后台登录页面



图 22.10　后台首页

管理员可以在后台查看图书销量排行，效果如图 22.11 所示，还可以对商城订单进行处理，效果如图 22.12 所示。



图 22.11　销量排行榜页面



图 22.12　订单管理模块首页

22.3.5　文件夹组织结构

在编写代码之前，可以把系统中可能用到的文件夹先创建出来（例如，创建一个名为 images 的文件夹，用于保存网站中所使用的图片），这样不但可以方便以后的开发工作，也可以规范网站的整体架构。笔者在开发图书商城时，设计了图 22.13 所示的文件夹架构图。在开发时，只需要将所创建的文件保存在相应的文件夹中就可以了。

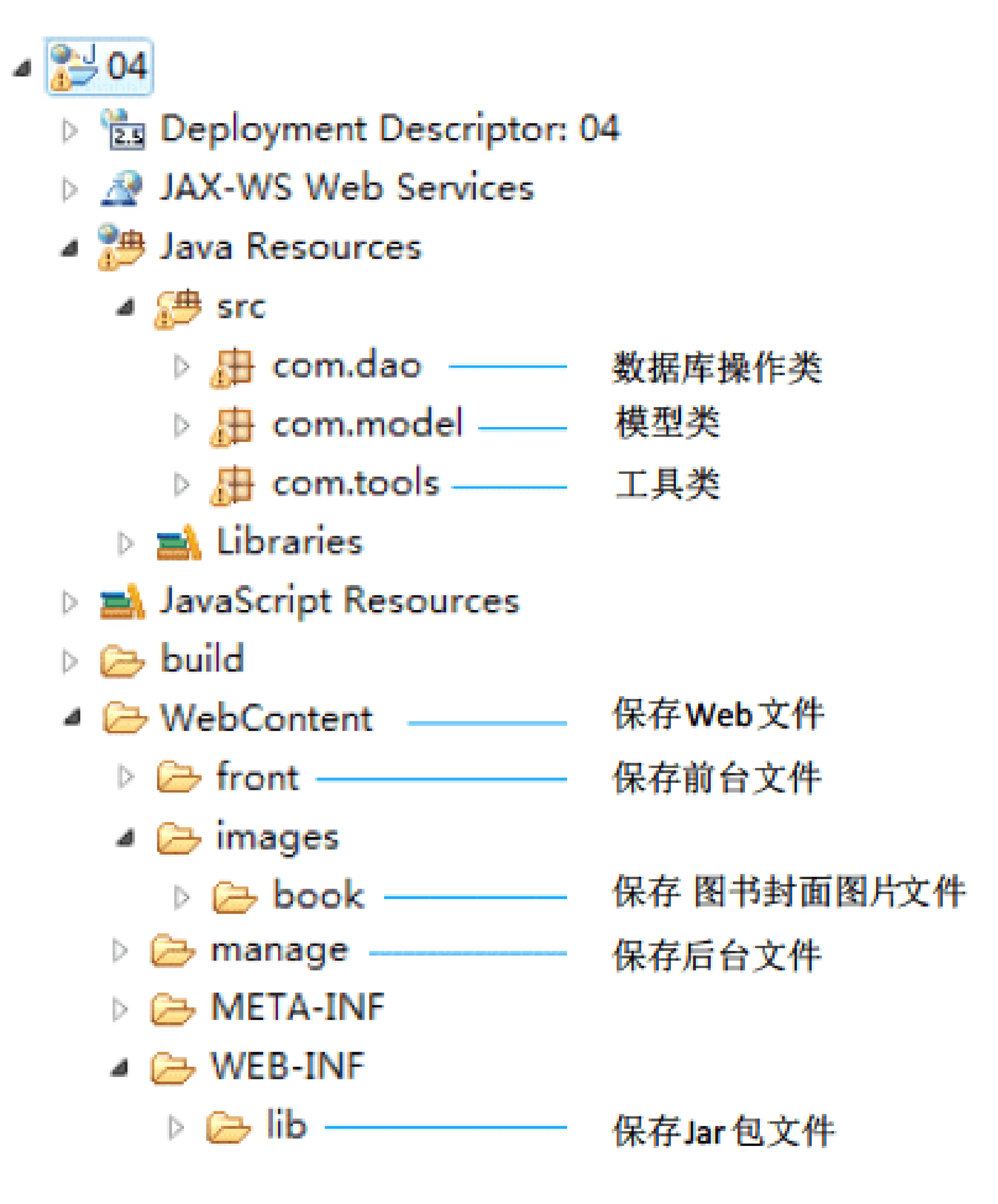


图 22.13　图书商城的文件夹架构图

22.4　数据库设计

22.4.1　数据库分析

为防止数据访问量增加使系统资源不足而导致系统崩溃，本程序采用了独立 SQL Server 数据服务器，将数据库单独放在一个服务器中。这样即使服务器系统崩溃了，数据库服务器也不会受到影响；



而且能够更快、更好地处理更多的数据。其数据库运行环境如下。

☑　硬件平台

CPU：P4 3.2GHz。

内存：2GB 以上。

硬盘空间：160GB。

☑　软件平台

操作系统：Windows 7 以上。

数据库：SQL Server 2014。

22.4.2　数据库概念设计

图书商城使用的是 SQL Server 2014 数据库，数据库名称为 db_book，共用到了 7 张数据表，其结构如图 22.14 所示。

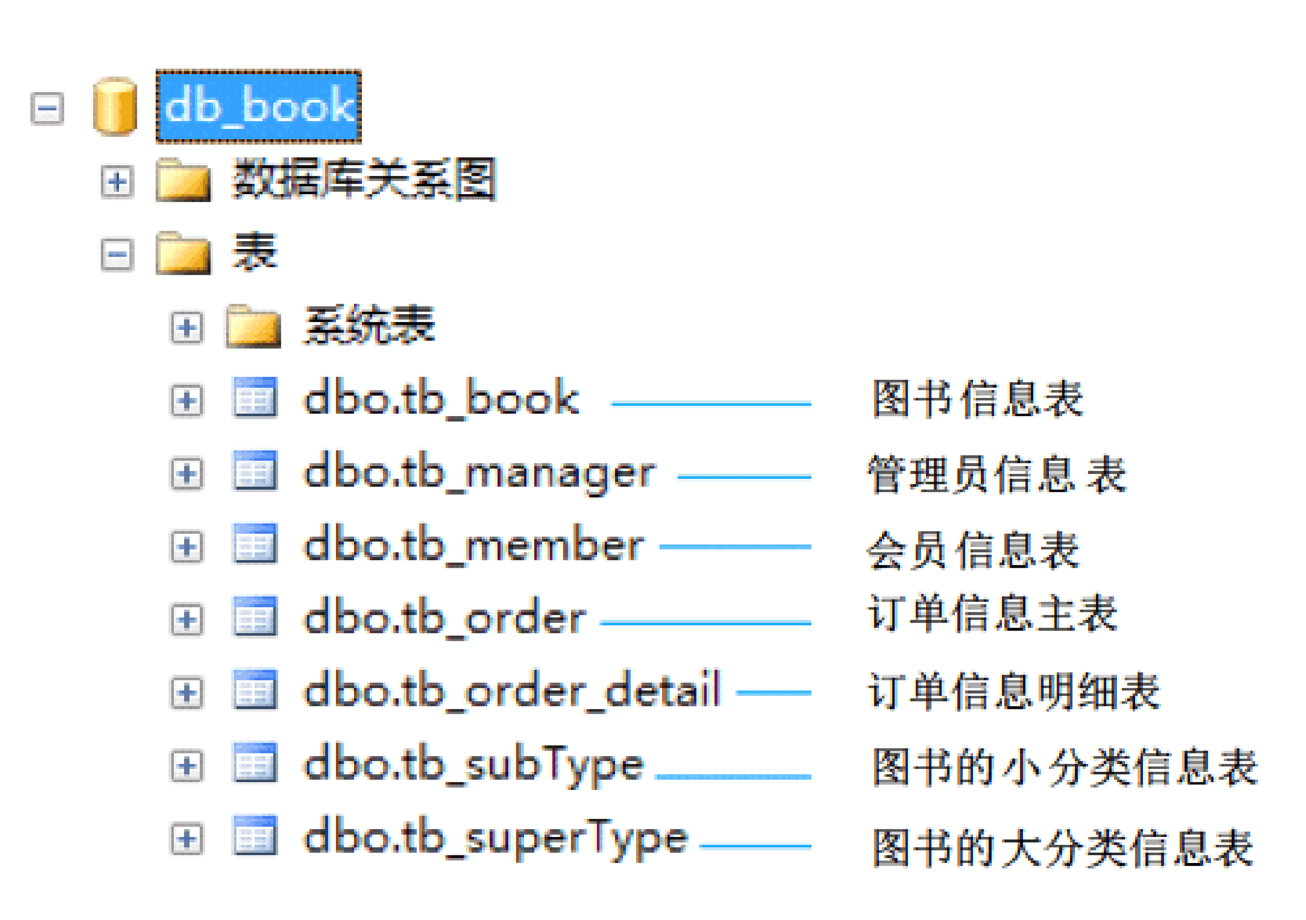


图 22.14　图书商城的数据库结构图

22.4.3　数据库逻辑结构设计

图书商城的各数据表的结构如下。

☑　会员信息表

会员信息表（tb_member）主要用来保存注册的会员信息，其结构如表 22.1 所示。

表 22.1　tb_member 结构

字　段　名	数 据 类 型	是否 Null 值	默认值或绑定	描　　述
ID	int(4)	No		ID（自动编号）
userName	varchar(20)	Yes		账户
trueName	varchar(20)	Yes		真实姓名
passWord	varchar(20)	Yes		密码
city	varchar(20)	Yes		城市
address	varchar(100)	Yes		地址
postcode	varchar(6)	Yes		邮编

续表

字 段 名	数 据 类 型	是否 Null 值	默认值或绑定	描 述
cardNO	varchar(24)	Yes		证件号码
cardType	varchar(20)	Yes		证件类型
grade	int(4)	Yes	0	等级
Amount	money	Yes	0	消费金额
tel	varchar(20)	Yes		联系电话
email	varchar(100)	Yes		E-mail
freeze	int(4)	Yes	0	是否冻结

☑ 图书的大分类信息表

大分类信息表（tb_superType）主要用来保存图书的大分类信息，也就是父分类，其结构如表 22.2 所示。

表 22.2 tb_superType 结构

字 段 名	数 据 类 型	是否 Null 值	默认值或绑定	描 述
ID	int	No		ID 号
TypeName	varchar(50)	No		分类名称

☑ 图书的小分类信息表

小分类信息表（tb_subType）主要用来保存图书的小分类信息，也就是子分类，其结构如表 22.3 所示。

表 22.3 tb_subType 结构

字 段 名	数 据 类 型	是否 Null 值	默认值或绑定	描 述
ID	int	No		ID 号
superType	int	No		父类 ID 号
TypeName	varchar(50)	No		分类名称

☑ 图书信息表

图书信息表（tb_book）主要用来保存图书信息，其结构如表 22.4 所示。

表 22.4 tb_book 结构

字 段 名	数 据 类 型	是否 Null 值	默认值或绑定	描 述
ID	bigint	No		图书 ID
typeID	int	No		类别 ID
bookName	varchar(200)	No		图书名称
introduce	text	Yes		图书简介
price	money	No		定价
nowPrice	money	Yes		现价
picture	varchar(100)	Yes		图片文件
INTime	datetime	Yes	getdate()	录入时间
newBook	int	No	0	是否新书，1 为是，默认 0



续表

字　段　名	数 据 类 型	是否 Null 值	默认值或绑定	描　　述
sale	int	Yes	0	是否特价，1 为是，默认 0
hit	int	Yes	0	浏览次数

☑　订单信息主表

订单信息主表（tb_order）用来保存订单的概要信息，其结构如表 22.5 所示。

表 22.5　tb_order 结构

字　段　名	数 据 类 型	是否 Null 值	默认值或绑定	描　　述
OrderID	bigint	No		订单编号
bnumber	smallint	Yes		品种数
username	varchar(15)	Yes		用户名
recevieName	varchar(15)	Yes		收货人
address	varchar(100)	Yes		收货地址
tel	varchar(20)	Yes		联系电话
OrderDate	smalldatetime	Yes	(getdate())	订单日期
bz	varchar(200)	Yes		备注

☑　订单信息明细表

订单信息明细表（tb_order_detail）用来保存订单的详细信息，其结构如表 22.6 所示。

表 22.6　tb_order_detail 结构

字　段　名	数 据 类 型	是否 Null 值	默认值或绑定	描述
ID	bigint	No		ID 号
orderID	bigint	No		与 tb_order 表的 OrderID 字段关联
bookID	bigint	No		图书 ID
price	money	No		价格
number	int	No		数量

☑　管理员信息表

管理员信息表（tb_manager）用来保存管理员信息，其结构如表 22.7 所示。

表 22.7　tb_manager 结构

字　段　名	数 据 类 型	是否 Null 值	默认值或绑定	描　　述
ID	int	No		ID 号
manager	varchar(30)	No		管理员名称
PWD	varchar(30)	No		密码

22.5　公共类设计

在开发程序时，经常会遇到在不同的方法中进行相同处理的情况，例如数据库连接和字符串处理

等，为了避免重复编码，可将这些处理封装到单独的类中，通常称这些类为公共类或工具类。在开发本网站时，用到以下公共类：数据库连接及操作类和字符串处理类，下面分别进行介绍。

22.5.1 数据库连接及操作类的编写

数据库连接及操作类通常包括连接数据库的方法 getConnection、执行查询语句的方法 executeQuery、执行更新操作的方法 executeUpdate 和关闭数据库连接的方法 close。下面将详细介绍如何编写图书商城的数据库连接及操作的类 ConnDB。

（1）创建用于进行数据库连接及操作的类 ConnDB，并将其保存到 com.mingrisoft.core 包中，同时定义该类中所需要的全局变量，在这里会指定数据库驱动类的类名、连接数据库的 URL 地址、登录 SQL Server 的用户名和密码等，代码如下：

```
package com.tools;
public class ConnDB {
	public Connection conn = null;				//数据库连接对象
	public Statement stmt = null;				//Statement 对象，用于执行 SQL 语句
	public ResultSet rs = null;				//结果集对象
	//驱动类的类名
	private static String dbClassName = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
	private static String dbUrl="jdbc:sqlserver://127.0.0.1:1433;DatabaseName=db_book";
	private static String dbUser = "sa";			//登录 SQL Server 的用户名
	private static String dbPwd = "";				//登录 SQL Server 的密码
}
```

（2）创建连接数据库的方法 getConnection，用于根据指定的数据库驱动获取数据库连接对象，如果连接失败，则输出异常信息。该方法返回一个数据库连接对象。getConnection 方法的具体代码如下：

```
public static Connection getConnection() {
		Connection conn = null;				// 声明数据库连接对象
		try {						// 捕捉异常
			Class.forName(dbClassName).newInstance();  // 装载数据库驱动
			conn = DriverManager.getConnection(dbUrl, dbUser, dbPwd);// 获取数据库连接对象
		} catch (Exception ee) {			// 处理异常
			ee.printStackTrace();			// 输出异常信息
		}
		if (conn == null) {
			System.err.println("DbConnectionManager.getConnection():"
					+ dbClassName + "\r\n :" + dbUrl + "\r\n " + dbUser + "/"
					+ dbPwd);			// 输出连接信息，方便调试
		}
		return conn;					// 返回数据库连接对象
	}
```

（3）编写查询数据的方法 executeQuery。在该方法中，首先调用 getConnection 方法获取数据库连接对象，然后通过该对象的 createStatement 方法创建一个 Statement 对象，并且调用该对象的 executeQuery 方法执行指定的 SQL 语句，从而实现查询数据的功能。具体代码如下：



```
public ResultSet executeQuery(String sql) {
        try {                                              // 捕捉异常
            conn = getConnection(); //调用 getConnection 方法构造 Connection 对象的一个实例 conn
            stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
            rs = stmt.executeQuery(sql);
        } catch (SQLException ex) {
            System.err.println(ex.getMessage());           // 输出异常信息
        }
        return rs;                                         // 返回结果集对象
    }
```

（4）编写执行更新数据的方法 executeUpdate，返回值为 int 型的整数，代表更新的行数。executeQuery 方法的代码如下：

```
public int executeUpdate(String sql) {
    int result = 0;                                    // 定义保存更新行数的变量
    try {                                              // 捕捉异常
        conn = getConnection(); //调用 getConnection 方法构造 Connection 对象的一个实例 conn
        stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY);
        result = stmt.executeUpdate(sql);              // 执行更新操作
    } catch (SQLException ex) {
        result = 0;                                    // 将保存更新行数的变量赋值为 0，表示更新失败
    }
    return result;                                     // 返回保存更新行数的变量
}
```

（5）编写用于实现更新数据后获取生成的自动编号的 executeUpdate_id 方法，在该方法中，首先获取数据库连接对象，然后执行 SQL 语句插入一条数据，再执行一条特定的 SQL 语句，用于获取刚刚生成的自动编号，最后返回获取的结果。executeUpdate_id 方法的具体代码如下：

```
public int executeUpdate_id(String sql) {
    int result = 0;
    try {                                              // 捕捉异常
        conn = getConnection();                        // 获取数据库连接
        // 创建用于执行 SQL 语句的 Statement 对象
        stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
              ResultSet.CONCUR_READ_ONLY);
        result = stmt.executeUpdate(sql);              // 执行 SQL 语句
        String ID = "select @@IDENTITY as id";         // 定义用于获取刚刚生成的自动编号的 SQL 语句
        rs = stmt.executeQuery(ID);                    // 获取刚刚生成的自动编号
        if (rs.next()) {                               // 如果存在数据
            int autoID = rs.getInt("id");              // 把获取到的自动编号保存到变量 autoID 中
            result = autoID;
        }
    } catch (SQLException ex) {                        // 处理异常
        result = 0;
    }
```

```
        return result;                                  // 返回获取结果
}
```

（6）编写关闭数据库连接的方法 close。在该方法中，首先关闭结果集对象，然后关闭 Statement 对象，最后再关闭数据库连接对象。具体代码如下：

```
public void close() {
        try {                                           // 捕捉异常
                if (rs != null) {                       // 当 ResultSet 对象的实例 rs 不为空时
                        rs.close();                     // 关闭 ResultSet 对象
                }
                if (stmt != null) {                     // 当 Statement 对象的实例 stmt 不为空时
                        stmt.close();                   // 关闭 Statement 对象
                }
                if (conn != null) {                     // 当 Connection 对象的实例 conn 不为空时
                        conn.close();                   // 关闭 Connection 对象
                }
        } catch (Exception e) {
                e.printStackTrace(System.err);          // 输出异常信息
        }
}
```

22.5.2 字符串处理类

字符串处理的 JavaBean 是解决程序中经常出现的字符串处理问题的类。它包括两个方法：一个是将数据库和页面中有中文问题的字符串进行正确的显示和存储的方法 chStr；另一个是将字符串中的回车换行、空格及 HTML 标签正确显示的方法 convertStr。下面将详细介绍如何编写图书商城中的字符串处理的 JavaBean ChStr。

（1）编写解决输出中文乱码问题的方法 chStr，这里主要是指定的字符串转换为 UTF-8 编码。由于默认的 ISO-8859-1 不支持中文，所以需要转换为 UTF-8 编码。ChStr 的具体代码如下：

```
public class ChStr {
        public String chStr(String str) {
                if (str == null) {                                              // 当变量 str 为 null 时
                        str = "";                                               // 将变量 str 赋值为空
                } else {
                        try {                                                   // 捕捉异常
                                str = (new String(str.getBytes("iso-8859-1"), "GBK")).trim();  // 将字符串转换为 GBK 编码
                        } catch (Exception e) {                                 // 处理异常
                                e.printStackTrace(System.err);                  // 输出异常信息
                        }
                }
                return str;                                                     // 返回转换后的变量 str
        }
        public String convertStr(String str1) {
                if (str1 == null) {
                        str1 = "";
```

```
            } else {
                try {
                    str1 = str1.replaceAll("<", "&lt;");// 替换字符串中的"<"和">"字符，保证 HTML 标记的正常输出
                    str1 = str1.replaceAll(">", "&gt;");
                    str1 = str1.replaceAll(" ", " ");
                    str1 = str1.replaceAll("\r\n", "<br>");
                } catch (Exception e) {
                    e.printStackTrace(System.err);
                }
            }
            return str1;
        }
}
```

（2）编写显示文本中的回车换行、空格及保证 HTML 标签的正常输出的方法 convertStr，这里主要是为了解决显示字符串内容时，HTML 标签中的字符将被作为 HTML 标签被浏览器解析，而不是原样显示的问题。convertStr 方法的代码如下：

```
public static String convertStr(String source){
    String changeStr="";
    changeStr=source.replaceAll("&","&amp;");              //转换字符串中的“&”符号
    changeStr=changeStr.replaceAll(" "," ");           //转换字符串中的空格
    changeStr=changeStr.replaceAll("<","&lt;");             //转换字符串中的“<”符号
    changeStr=changeStr.replaceAll(">","&gt;");             //转换字符串中的“>”符号
    changeStr=changeStr.replaceAll("\r\n","<br>");          //转换字符串中的回车换行
    return changeStr;
}
```

22.6　会员注册模块设计

22.6.1　会员注册模块概述

会员注册页面主要对网站的用户信息进行注册，包括登录账户、真实姓名、密码、联系电话和邮箱等。运行结果请参照 22.6.4 节中的图 22.17。

22.6.2　创建会员对应的模型类 Member

创建会员对应的模型类 Member，将该类保存到 com.model 包中。创建模型类的具体方法如下。

（1）在 com.model 中创建一个名称为 Member 的 Java 类，然后在该类中创建一些属性，这些属性通常是与会员信息表的字段相对应的，代码如下：

```
public class Member {
```

```
    private Integer ID = Integer.valueOf("-1");         // 定义会员 ID 属性
    private String username = "";                      // 定义账户属性
    private String truename = "";                      // 定义真实姓名属性
    private String pwd = "";                           // 定义密码属性
    private String city = "";                          // 定义所在城市属性
    private String address = "";                       // 定义地址属性
    private String postcode = "";                      // 定义邮编属性
    private String cardno = "";                        // 定义证件号码属性
    private String cardtype = "";                      // 定义证件类型属性
    private String tel = "";                           // 定义联系电话属性
    private String email = "";                         // 定义邮箱属性
}
```

（2）在 Member.java 文件中，为各个属性创建对应的赋值方法和获取值的方法，具体方法如下。

第一步：在页面中最后一个“}”之前单击鼠标右键，在弹出的快捷菜单中选择 Source/Generate Getters and Setters 菜单项，如图 22.15 所示。

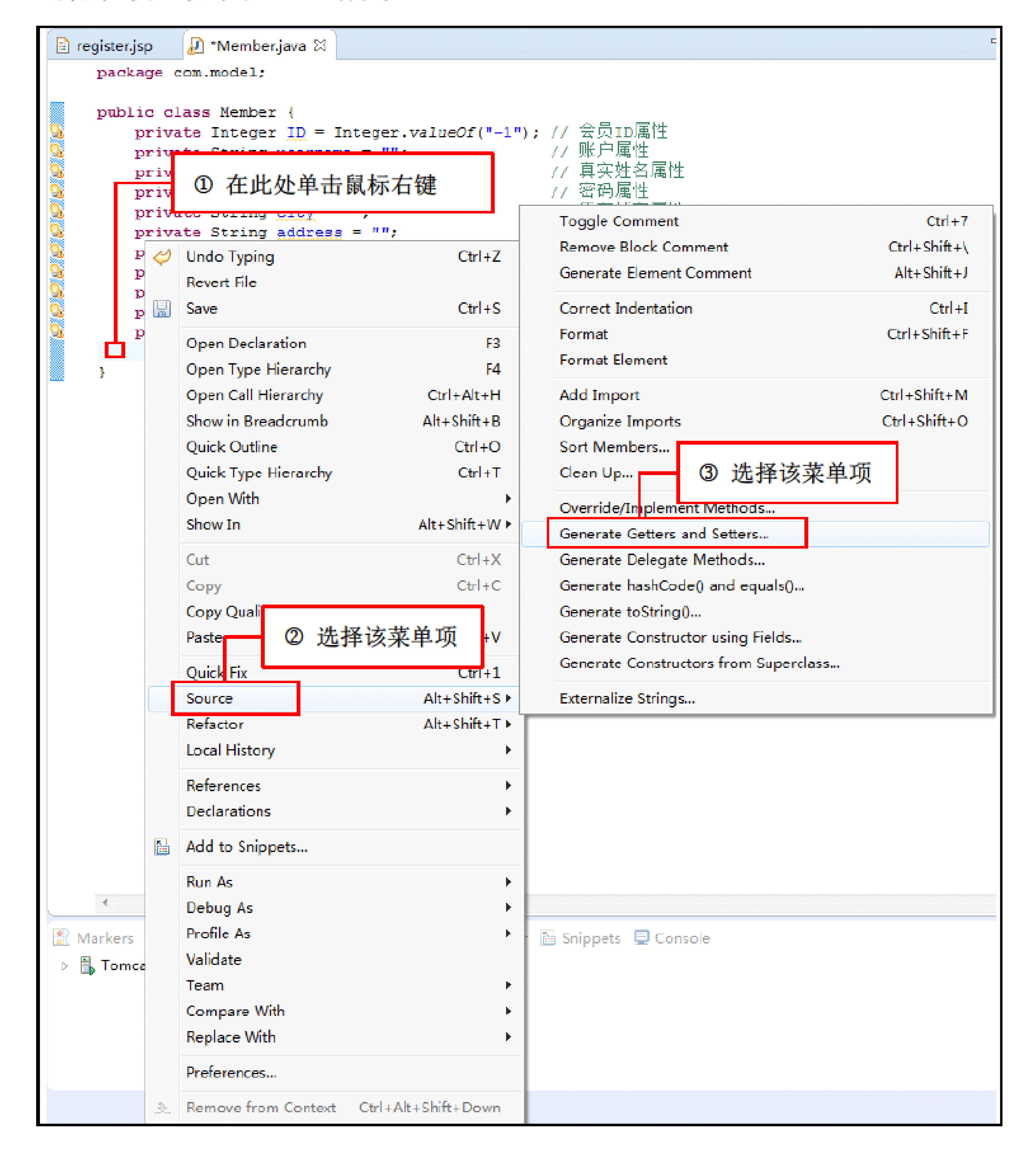


图 22.15　创建赋值方法和获取值的方法



第二步：在打开的 Generate Getters and Setters 对话框中，选中全部复选框，其他采用默认，如图 22.16 所示。

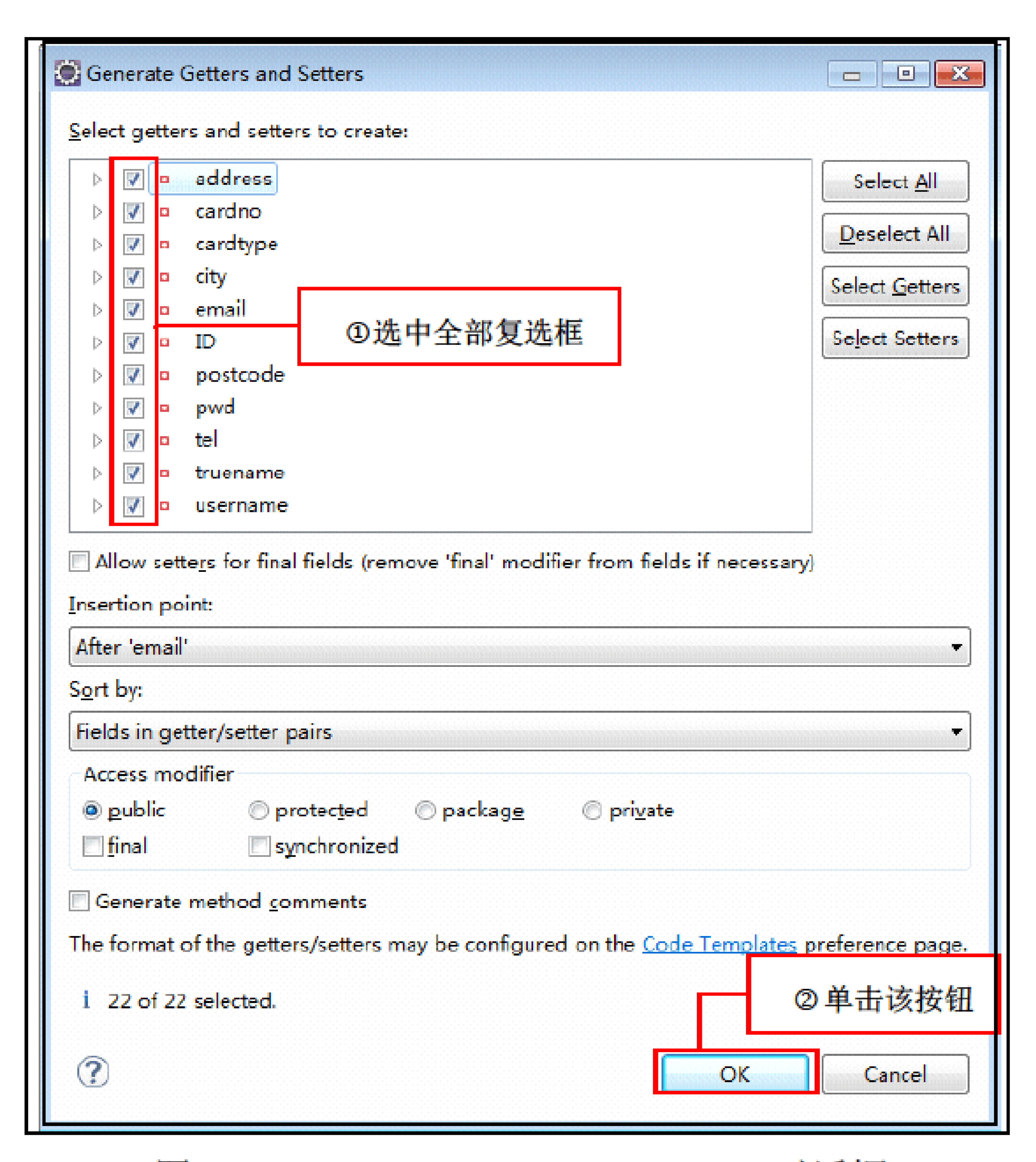


图 22.16　Generate Getters and Setters 对话框

（3）按下快捷键 Ctrl+S 保存文件。此时，Member 类就创建完毕了

22.6.3　创建会员对应的数据库操作类

创建会员对应的数据库操作类，位于 com.dao 包中。主要通过创建并实现接口来完成的，具体步骤如下。

（1）在 com.dao 包中创建一个名称为 MemberDao 的接口，并且在该接口的接口体中定义一个 insert 方法（用于保存会员信息）和一个 select 方法（用于查询会员信息）。需要注意的是，这里只进行方法的定义，没有具体的实现。具体代码如下：

```
import java.util.List;                          //导入 List 类
import com.model.Member;                        //导入会员模型类
public interface MemberDao {
    public int insert(Member m);                // 保存会员信息
    public List select();                       // 查询会员信息
}
```

（2）创建接口后，还必须实现该接口。在 com.dao 包上创建一个 MemberDao 接口的实现类，名称为 MemberDaoImpl，此时 Eclipse 会自动添加要实现的 inset 和 select 两个接口方法。自动生成的代

码如下：

```
import java.util.List;
import com.model.Member;
public class MemberDaoImpl implements MemberDao {
        @Override
        public int insert(Member m) {
                // TODO Auto-generated method stub
                return 0;
        }
        @Override
        public List select() {
                // TODO Auto-generated method stub
                return null;
        }
}
```

（3）在 MemberDaoImpl 类中，声明两个成员变量，用于创建数据库连接类的对象和字符串操作类的对象。这是由于在 Java 中想要使用类，必须先创建它的对象。关键代码如下：

```
private ConnDB conn = new ConnDB();                    // 创建数据库连接类的对象
private ChStr chStr = new ChStr();                     // 创建字符串操作类的对象
```

（4）在自动生成的 insert 方法中，编写向数据库保存会员信息的代码。这里主要是通过 SQL 语言中的 INSET INTO 语句实现向数据库中保存数据的。在执行完插入操作后，不要忘记关闭数据库的连接。代码如下：

```
public int insert(Member m) {
int ret = -1;                                          // 用于记录更新记录的条数
try {                                                  // 捕捉异常
        String sql = "Insert into tb_Member (UserName,TrueName,PassWord,City,address,postcode,"
                + "CardNO,CardType,Tel,Email) values('"
                + chStr.chStr(m.getUsername()) + "','" + chStr.chStr(m.getTruename()) + "','"
                + chStr.chStr(m.getPwd()) + "','" + chStr.chStr(m.getCity()) + "','"
                + chStr.chStr(m.getAddress())
                + "','" + chStr.chStr(m.getPostcode()) + "','" + chStr.chStr(m.getCardno())
                + "','"+ chStr.chStr(m.getCardtype()) + "','" + chStr.chStr(m.getTel()) + "','"
                + chStr.chStr(m.getEmail())
                + "')";                                // 用于实现保存会员信息的 SQL 语句
        ret = conn.executeUpdate(sql);                 // 执行 SQL 语句实现保存会员信息到数据库
} catch (Exception e) {                                // 处理异常
                e.printStackTrace();                   // 输出异常信息
                ret = 0;                               // 设置变量的值为 0，表示保存会员信息失败
}
conn.close();                                          // 关闭数据库的连接
return ret;
}
```

（5）在自动生成的 select 方法中，编写从数据库查询会员信息的代码。这里主要是通过数据库连接类的对象的 executeQuery 方法执行一条执行查询操作的 SQL 语句实现的。另外还需要把查询结果保存到 List 集合对象中，方便以后使用。具体代码如下：

```
public List select() {
Member form = null;                                          // 声明会员对象
List list = new ArrayList();                                 // 创建一个 List 集合对象，用于保存会员信息
String sql = "select * from tb_member";                      // 查询全部会员信息的 SQL 语句
ResultSet rs = conn.executeQuery(sql);                       // 执行查询操作
try {                                                        // 捕捉异常
        while (rs.next()) {
            form = new Member();                             // 实例化一个会员对象
            form.setID(Integer.valueOf(rs.getString(1)));    // 获取会员 ID
            list.add(form);                                  // 把会员信息添加到 List 集合对象中
        }
} catch (SQLException ex) {                                  // 处理异常
}
conn.close();                                                // 关闭数据库的连接
return list;
}
```

22.6.4　设计会员注册页面

设计一个名称为 register.jsp 的首页，在该页面中主要通过 HTML 和 CSS 实现一个图 22.17 所示的静态页面。在该页面中，最核心的代码就是用于收集会员注册信息的表单及表单元素。



图 22.17　静态的会员注册页面

22.6.5 实现保存会员信息页面

在实现会员注册时，需要给表单设置一个处理页，用来保存会员的注册信息。本项目中采用一个名称为 register_deal.jsp 的 JSP 文件作为处理页，该文件的具体实现步骤如下。

（1）在项目的 WebContent/front 节点中，创建一个名称为 register_deal.jsp 的 JSP 文件，在该文件中分别创建 ConnDB、MemberDaoImpl 和 Member 类的对象，并且通过<jsp:setProperty name="member" property="*"/>对 Member 类的所有属性进行赋值，用于获取用户填写的注册信息，关键代码如下：

```
<%-- 创建 ConnDB 类的对象 --%>
<jsp:useBean id="conn" scope="page" class="com.tools.ConnDB" />
<%-- 创建 MemberDaoImpl 类的对象 --%>
<jsp:useBean id="ins_member" scope="page" class="com.dao.MemberDaoImpl" />
<%-- 创建 Member 类的对象，并对 Member 类的所有属性进行赋值 --%>
<jsp:useBean id="member" scope="request" class="com.model.Member">
    <jsp:setProperty name="member" property="*" />
</jsp:useBean>
```

（2）判断输入的账号是否存在，如果存在给予提示，否则调用 MemberDaoImpl 类的 insert 方法，将填写的会员信息保存到数据库中。具体代码如下：

```
<%
    request.setCharacterEncoding("UTF-8");                      //设置请求的编码为 UTF-8
    String username = member.getUsername();                     //获取会员账号
    ResultSet rs = conn.executeQuery("select * from tb_Member where username='"
     + username + "'");
    if (rs.next()) {                                            //如果结果集中有数据
        out.println("<script language='javascript'>alert('该账号已经存在，请重新注册！');"
                + "window.location.href='register.jsp';</script>");
    } else {
        int ret = 0;                                            //记录更新记录条数的变量
        ret = ins_member.insert(member);                        //将填写的会员信息保存到数据库
        if (ret != 0) {
            session.setAttribute("username", username);        //将会员账号保存到 Session 中
            out.println("<script language='javascript'>alert('会员注册成功！');"
                    + "window.location.href='index.jsp';</script>");
        } else {
            out.println("<script language='javascript'>alert('会员注册失败！');"
                    + "window.location.href='register.jsp';</script>");
        }
    }
%>
```

运行程序，在会员注册页面中填写图 22.18 所示的会员信息，然后单击“注册”按钮，即可将该信息保存到数据库中，同时显示图 22.19 所示的提示框。





图 22.18　填写会员信息

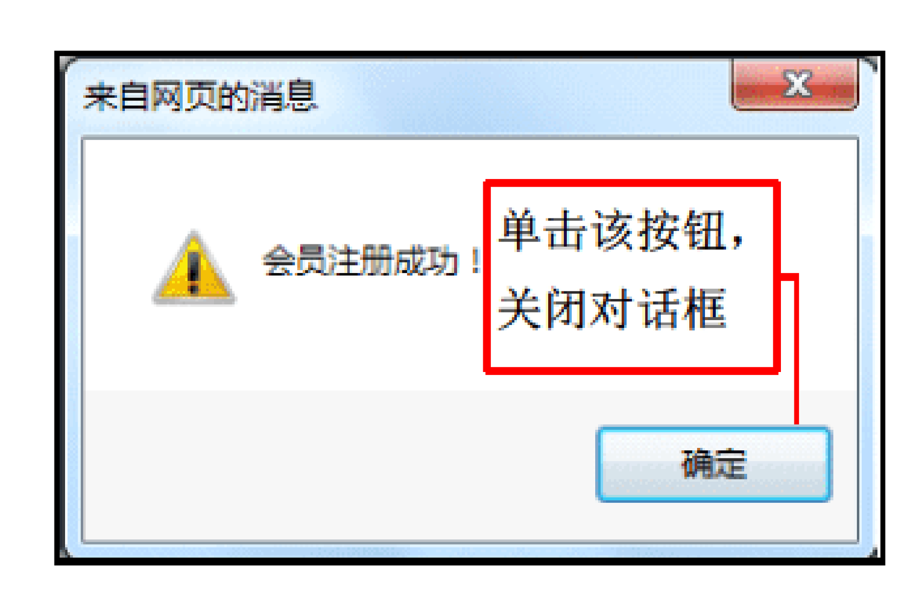


图 22.19　提示会员注册成功

22.7　会员登录模块设计

22.7.1　会员登录模块概述

会员登录模块主要用于实现网站的会员功能。在会员登录页面中，填写会员账户、密码和验证码（如果验证码看不清楚可以单击验证码图片刷新该验证码），如图 22.20 所示，单击“登录”按钮，即可实现会员登录。如果没有输入账户、密码或者验证码，都将给予提示。另外，验证码输入错误也将给予提示。



图 22.20　会员登录页面



22.7.2　设计会员登录页面

设计一个名称为 login.jsp 的页面，在该页面中主要通过 HTML 和 CSS 实现一个图 22.21 所示的静态页面。在该页面中，最核心的代码就是用于收集会员登录信息的表单及表单元素。

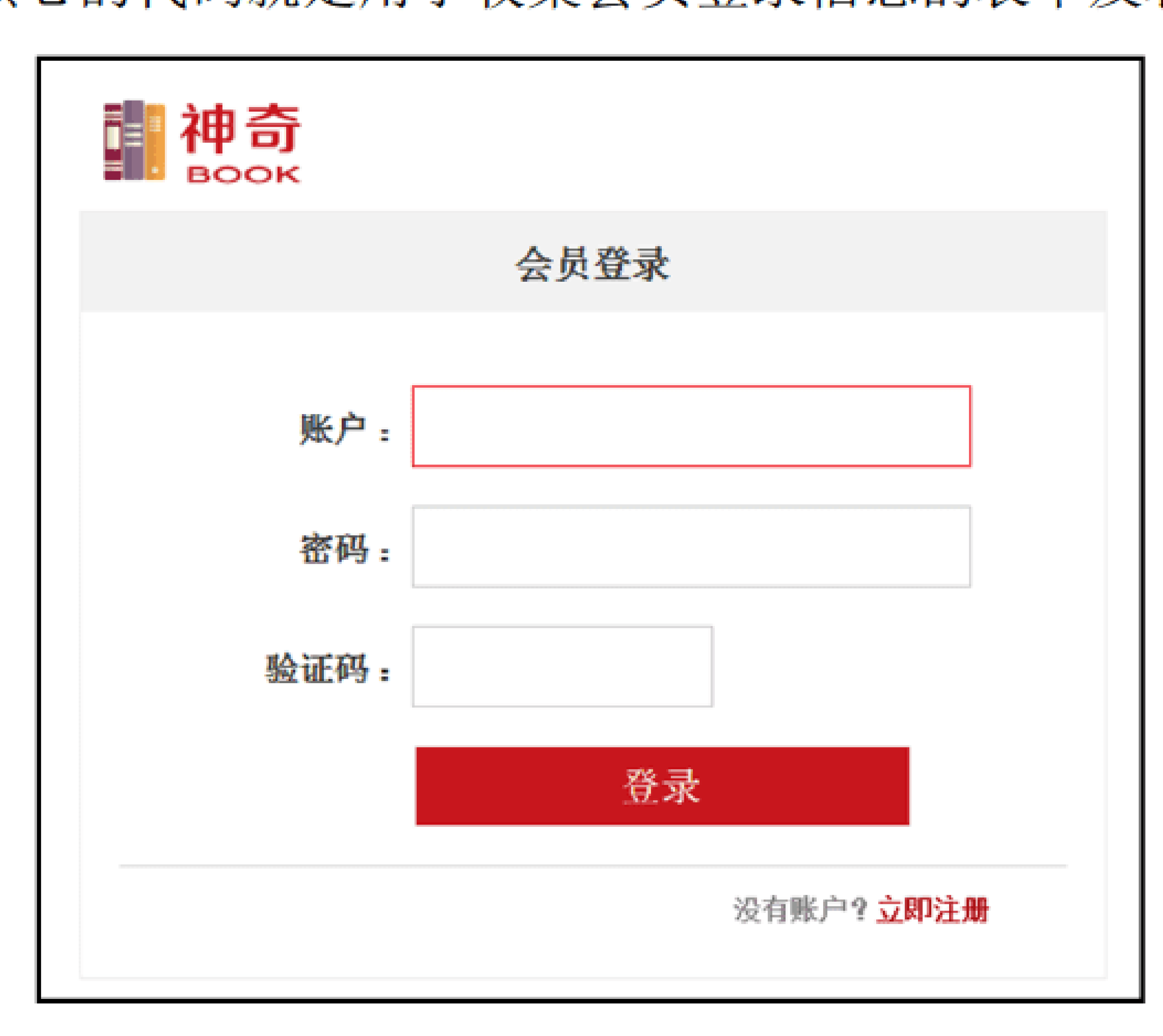


图 22.21　静态的会员登录页面

22.7.3　实现验证码

由于在图书商城的会员登录页面中，需要提供验证码功能，防止恶意登录，所以需要在会员登录页面中添加验证码，大致可以分为以下 3 个步骤。

（1）创建一个用于生成验证码的 Servlet，名称为 CheckCode.java。在该文件中通过 Java 的绘图类提供的方法生成带干扰线的随机验证码。关键步骤如下。

由于在生成验证码的过程中，需要随机生成输出内容的颜色，所以需要编写一个用于随机生成 RGB 颜色的方法，该方法的名称为 getRandColor，返回值为 java.awt.Color 类型的颜色。getRandColor 方法的具体代码如下：

```
// 获取随机颜色
public Color getRandColor(int s, int e) {
	Random random = new Random();
	if (s > 255) s = 255;
	if (e > 255) e = 255;
	int r = s + random.nextInt(e - s);			//随机生成 RGB 颜色中的 r 值
	int g = s + random.nextInt(e - s);			//随机生成 RGB 颜色中的 g 值
	int b = s + random.nextInt(e - s);			//随机生成 RGB 颜色中的 b 值
	return new Color(r, g, b);
}
```



在 service 方法中，设置响应头信息并指定生成的响应是 JPEG 图片，具体代码如下：

```
/**  禁止缓存**/
response.setHeader("Pragma", "No-cache");
response.setHeader("Cache-Control", "No-cache");
response.setDateHeader("Expires", 0);
/**********/
response.setContentType("image/jpeg");                    //指定生成的响应是图片
```

创建用于生成验证码的绘图类对象，并绘制一个填色矩形作为验证码的背景，具体代码如下：

```
int width = 116;                                          //指定验证码的宽度
int height = 33;                                          //指定验证码的高度
BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
Graphics g = image.getGraphics();                         //获取 Graphics 类的对象
Random random = new Random();                             //实例化一个 Random 对象
Font mFont = new Font("宋体", Font.BOLD, 22);             //通过 Font 构造字体
g.fillRect(0, 0, width, height);                          //绘制验证码背景
```

设置字体和颜色，随机绘制 100 条随机直线，具体代码如下：

```
g.setFont(mFont);                                         //设置字体
g.setColor(getRandColor(180, 200));                       //设置颜色
//  画随机的线条
for (int i = 0; i < 100; i++) {
      int x = random.nextInt(width - 1);
      int y = random.nextInt(height - 1);
      int x1 = random.nextInt(3) + 1;
      int y1 = random.nextInt(6) + 1;
    g.drawLine(x, y, x + x1, y + y1);                     //绘制直线
}
```

绘制一条折线，颜色为灰色，位置随机产生，线条粗细为 2f，具体代码如下：

```
//创建一个供画笔选择线条粗细的对象
BasicStro ke bs=new BasicStroke(2f,BasicStroke.CAP_BUTT,BasicStroke.JOIN_BEVEL);
Graphics2D g2d = (Graphics2D) g;                  //通过 Graphics 类的对象创建一个 Graphics2D 类的对象
g2d.setStroke(bs);                                //改变线条的粗细
g.setColor(Color.GRAY);                           //设置当前颜色为预定义颜色中的灰色
int lineNumber=4;                                 //指定端点的个数
int[] xPoints=new int[lineNumber];                //定义保存 x 轴坐标的数组
int[] yPoints=new int[lineNumber];                //定义保存 y 轴坐标的数组
//通过循环为 x 轴坐标和 y 轴坐标的数组赋值
for(int j=0;j<lineNumber;j++){
      xPoints[j]=random.nextInt(width - 1);
      yPoints[j]=random.nextInt(height - 1);
}
g.drawPolyline(xPoints, yPoints,lineNumber);      //绘制折线
```

随机生成由 4 个英文字母组成的验证码文字，并对文字进行随机缩放并旋转，具体代码如下：

```
String sRand = "";
// 输出随机的验证文字
for (int i = 0; i < 4; i++) {
    char ctmp = (char)(random.nextInt(26) + 65);                     //生成 A~Z 的字母
      sRand += ctmp;
      Color color = new Color(20 + random.nextInt(110), 20 + random
                  .nextInt(110), 20 + random.nextInt(110));
      g.setColor(color);                                              //设置颜色
      /** **随机缩放文字并将文字旋转指定角度* */
      // 将文字旋转指定角度
      Graphics2D g2d_word = (Graphics2D) g;
      AffineTransform trans = new AffineTransform();
      trans.rotate(random.nextInt(45) * 3.14 / 180, 22 * i + 8, 7);
      // 缩放文字
      float scaleSize = random.nextFloat() +0.8f;
      if (scaleSize > 1f)     scaleSize = 1f;
      trans.scale(scaleSize, scaleSize);                              //进行缩放
      g2d_word.setTransform(trans);
      /** ******************** */
      g.drawString(String.valueOf(ctmp), width/6 * i+23, height/2);   //绘制字符串
}
```

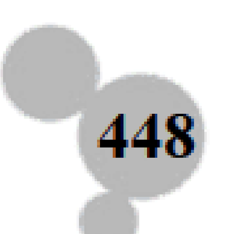
将生成的验证码保存到 Session 中，并输出生成后的验证码图片，具体代码如下：

```
/** 将生成的验证码保存到 Session 中***/
HttpSession session = request.getSession(true);
session.setAttribute("randCheckCode", sRand);
/****************************/
g.dispose();                                                          //销毁绘图类的对象
ImageIO.write(image, "JPEG", response.getOutputStream());             //指定图片的格式为 JPEG
```

（2）打开 book/WebContent/WEB-INF/web.xml 文件，在该文件中配置生成验证码的 Servlet。在配置该 Servlet 时，主要是通过<servlet>标记先配置 Servlet 文件，然后再通过<servlet-mapping>标记配置一个映射路径，用于使用该 Servlet。关键代码如下：

```
<servlet>
      <servlet-name>CheckCode</servlet-name>
      <servlet-class>com.tools.CheckCode</servlet-class>
</servlet>
<servlet-mapping>
      <servlet-name>CheckCode</servlet-name>
      <url-pattern>/CheckCode</url-pattern>
</servlet-mapping>
```

（3）在会员登录页面 login.jsp 的验证码文本框的右侧插入以下代码，用于使用<img>标记显示验



证码，并且实现单击该验证码时重新获取一个验证码。

```
<img src="../CheckCode" name="img_checkCode" onClick="myReload()" width="116"
        height="43" class="img_checkcode" id="img_checkCode" />
```

在上面的代码中，onClick="myReload()"的作用是调用 myReload 方法，实现单击验证码图片时，重新获取一个验证码。

22.7.4　编写会员登录处理页

同会员注册模块一样，在实现会员登录时，也需要给表单设置一个处理页，该处理页用来将输入的账户和密码与数据库中的进行匹配，并给出提示。在本项目中，会员登录处理页名称为 login_check.jsp。创建 login_check.jsp 文件的具体步骤如下。

（1）在项目的 WebContent/front 节点下创建一个名称为 login_check.jsp 的 JSP 文件，并且在该文件中添加以下代码。用于导入 java.sql 包中的 ResultSet 类，并且创建 ConnDB 类的对象。

```
<%-- 导入 java.sql.ResultSet 类 --%>
<%@ page import="java.sql.ResultSet"%>
<%-- 创建 ConnDB 类的对象 --%>
<jsp:useBean id="conn" scope="page" class="com.tools.ConnDB" />
```

（2）获取输入的账号和密码，并将其与数据库中保存的账户和密码进行匹配，并且根据匹配结果给予相应的提示，并转到指定页面。具体代码如下：

```
<%
String username = request.getParameter("username");                    //获取账户
String checkCode = request.getParameter("checkCode");                  //获取验证码
if (checkCode.equals(session.getAttribute("randCheckCode").toString())) {
        try {                                                          //捕捉异常
                ResultSet rs = conn.executeQuery("select * from tb_Member where username='" + username + "'");
                if (rs.next()) {                                       //如果找到相应的账号
                        String PWD = request.getParameter("PWD");      //获取密码
                        if (PWD.equals(rs.getString("password"))) {    //如果输入的密码和获取的密码一致
                                //把当前的账户保存到 Session 中，实现登录
                                session.setAttribute("username", username);
                                response.sendRedirect("index.jsp");    //跳转到前台首页
                        } else {
                                out.println(
                                "<script language='javascript'>alert('您输入的用户名或密码错误，请与管理员联系!');"
                                                +"window.location.href='login.jsp';</script>");
                        }
                } else {
                        out.println(
                                "<script language='javascript'>alert('您输入的用户名或密码错误，或您的账户"+
                                "已经被冻结，请与管理员联系!');window.location.href='login.jsp';</script>");
```

```
            }
        } catch (Exception e) {                                    //处理异常
            out.println(
                    "<script language='javascript'>alert('您的操作有误!');"
                    +"window.location.href='login.jsp';</script>");
        }
        conn.close();                                              //关闭数据库连接
    } else {
        out.println("<script language='javascript'>alert('您输入的验证码错误!');history.back();</script>");
    }
%>
```

按下快捷键 Ctrl+S 保存文件。在地址栏中输入 http://localhost:8080/shop/front/login.jsp，并按下 Enter 键，将显示会员登录页面，在该页面中输入已经注册好的会员账户和密码，如图 22.22 所示，然后单击“登录”按钮，如果输入的会员账户和密码正确，则直接转到前台首页 index.jsp 页面（由于暂时还没有编写该页面，所以会显示图 22.23 所示的效果），否则给出相应的提示。

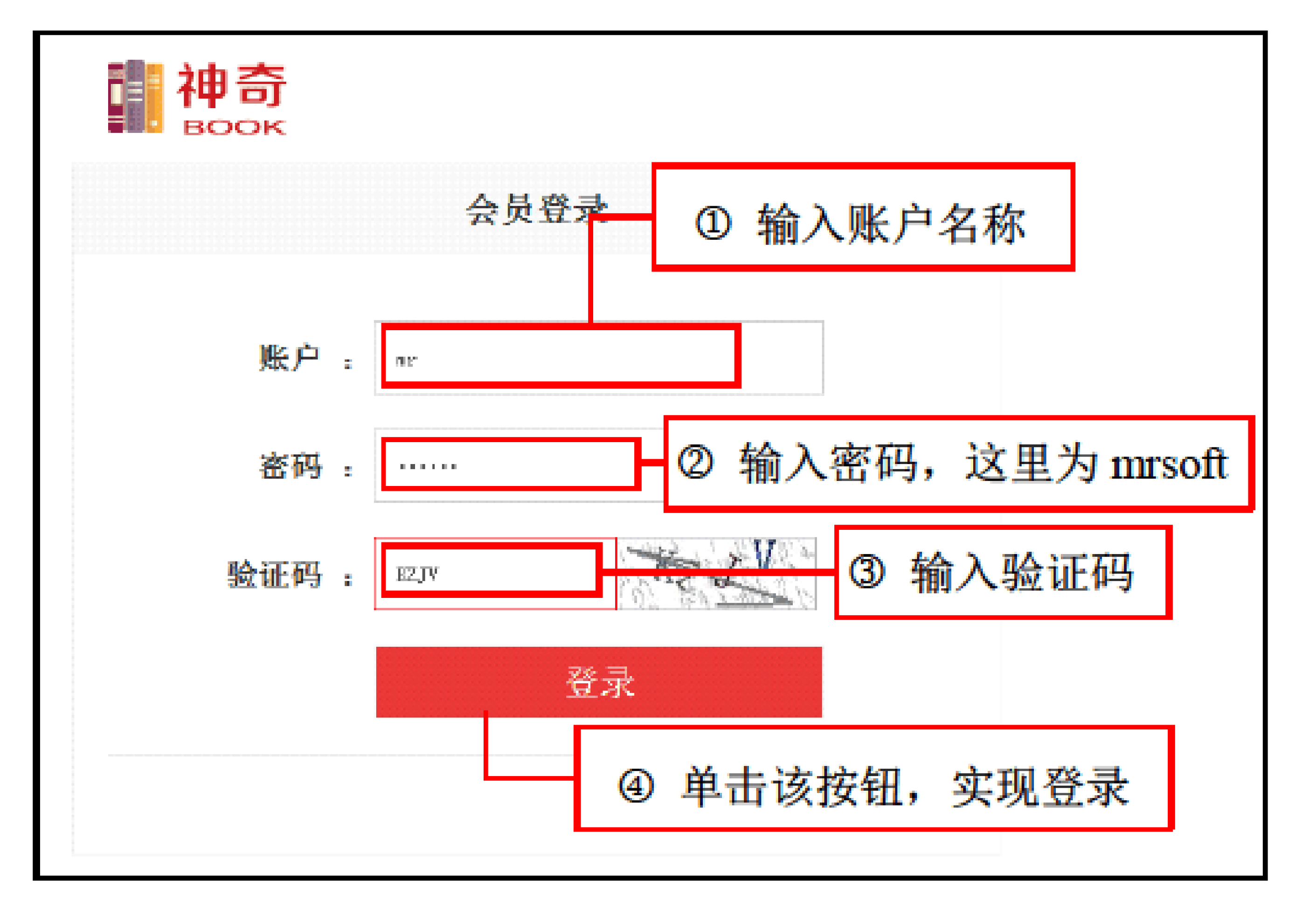


图 22.22　填写登录信息

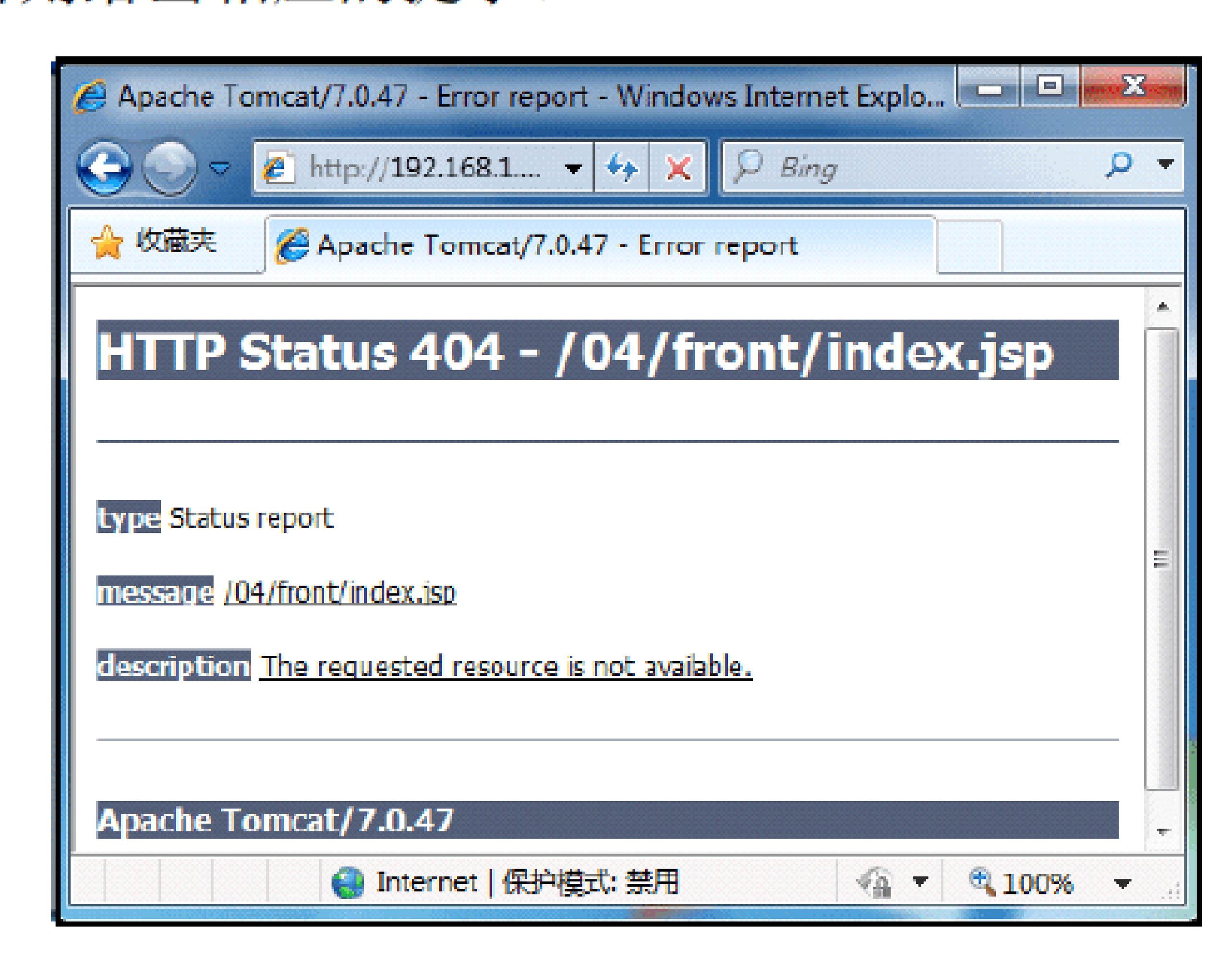


图 22.23　登录成功

22.8　首页模块设计

22.8.1　首页模块概述

当用户访问图书商城时，首先进入的便是前台“首页”。前台首页设计的美观程度将直接影响用户的购买欲望。在图书商城的前台首页中，用户不但可以查看最新上架、打折图书等信息，还可以及时了解大家喜爱的热门图书，以及商城推出的最新活动或者广告。图书商城前台首页的运行结果如图 22.24 所示。

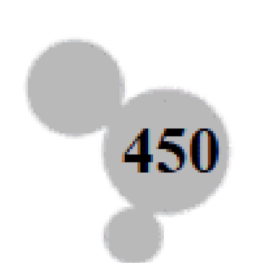




图 22.24　首页运行效果

22.8.2　设计首页界面

设计一个名称为 index.jsp 的首页，在该页面中主要通过 HTML 和 CSS 实现一个图 22.25 所示的静态页面。



图 22.25　设计完成的首页

在打开的图书商城的首页中，主要有 3 个部分需要我们添加动态代码，也就是把图 22.25 所示的 3 个区域中的图书信息，通过 JSP 代码从数据库中读取，并应用循环显示在页面上。

22.8.3　实现显示最新上架图书功能

打开首页文件 index.jsp，然后在该文件中添加用于显示最新上架图书的代码，具体步骤如下。

（1）由于在实现查询最新上架图书时，需要访问数据库，所以需要导入 java.sql.ResultSet 类并创建 com.tools.ConnDB 类的对象，具体代码如下：

```
<%@ page import="java.sql.ResultSet"%>            <%-- 导入 java.sql.ResultSet 类 --%>
<%-- 创建 com.tools.ConnDB 类的对象 --%>
<jsp:useBean id="conn" scope="page" class="com.tools.ConnDB" />
```

（2）调用 ConnDB 类的 executeQuery 方法执行 SQL 语句，用于从数据表中查询最新上架图书。另外，还需要定义保存图书信息的变量。具体代码如下：

```
<%
     /* 最新上架图书信息 */
     ResultSet rs_new = conn.executeQuery(
               "select top 12 t1.ID, t1.BookName,t1.price,t1.picture,t2.TypeName "
               +"from tb_book t1,tb_subType t2 where t1.typeID=t2.ID and "
               +"t1.newBook=1 order by t1.INTime desc");          //查询最新上架图书信息
     int new_ID = 0;                                              //保存最新上架图书 ID 的变量
     String new_bookname = "";                                    //保存最新上架图书名称的变量
     float new_nowprice = 0;                                      //保存最新上架图书价格的变量
     String new_picture = "";                                     //保存最新上架图书图片的变量
     String typeName = "";                                        //保存图书分类的变量
%>
```

（3）将获取到的图书信息显示到页面的最新上架图书展示区，这里面需要设置一个 while 循环，用于循环获取并显示每一条图书信息，关键代码如下：

```
<%
while (rs_new.next()) {                                           //设置一个循环
     new_ID = rs_new.getInt(1);                                   //获取最新上架图书的 ID
     new_bookname = rs_new.getString(2);                          //获取最新上架图书的图书名称
     new_nowprice = rs_new.getFloat(3);                           //获取最新上架图书的价格
     new_picture = rs_new.getString(4);                           //获取最新上架图书的图片
     typeName = rs_new.getString(5);                              //获取最新上架图书的类别
%>
...          <!--此处省略了将获取到的图书信息显示到指定位置的代码-->
<% } %>
```

运行程序，在首页中将显示图 22.26 所示的最新上架图书。



图 22.26　显示最新上架图书

22.8.4 实现显示打折图书功能

在 index.jsp 文件中添加用于显示打折图书的代码，具体步骤如下。

（1）调用 ConnDB 类的 executeQuery 方法执行 SQL 语句，用于从数据表中查询打折图书，这里也需要编写一个连接查询的 SQL 语句。另外，还需要定义保存图书信息的变量。具体代码如下：

```
/* 打折图书信息 */
ResultSet rs_sale = conn.executeQuery(
        "select top 12 t1.ID, t1.BookName,t1.price,t1.nowPrice,t1.picture,t2.TypeName "
        +"from tb_book t1,tb_subType t2 where t1.typeID=t2.ID and t1.sale=1 "
        +"order by t1.INTime desc");            //查询打折图书信息
int sale_ID = 0;                                //保存打折图书 ID 的变量
String s_bookname = "";                         //保存打折图书名称的变量
float s_price = 0;                              //保存打折图书的原价格的变量
float s_nowprice = 0;                           //保存打折图书的打折后价格的变量
String s_introduce = "";                        //保存打折图书简介的变量
String s_picture = "";                          //保存打折图书图片的变量
```

（2）将获取到的图书信息显示到页面的打折图书展示区，具体方法同 22.8.3 节的显示最新上架图书基本相同，这里不再赘述。

运行程序，将显示图 22.27 所示的打折图书。

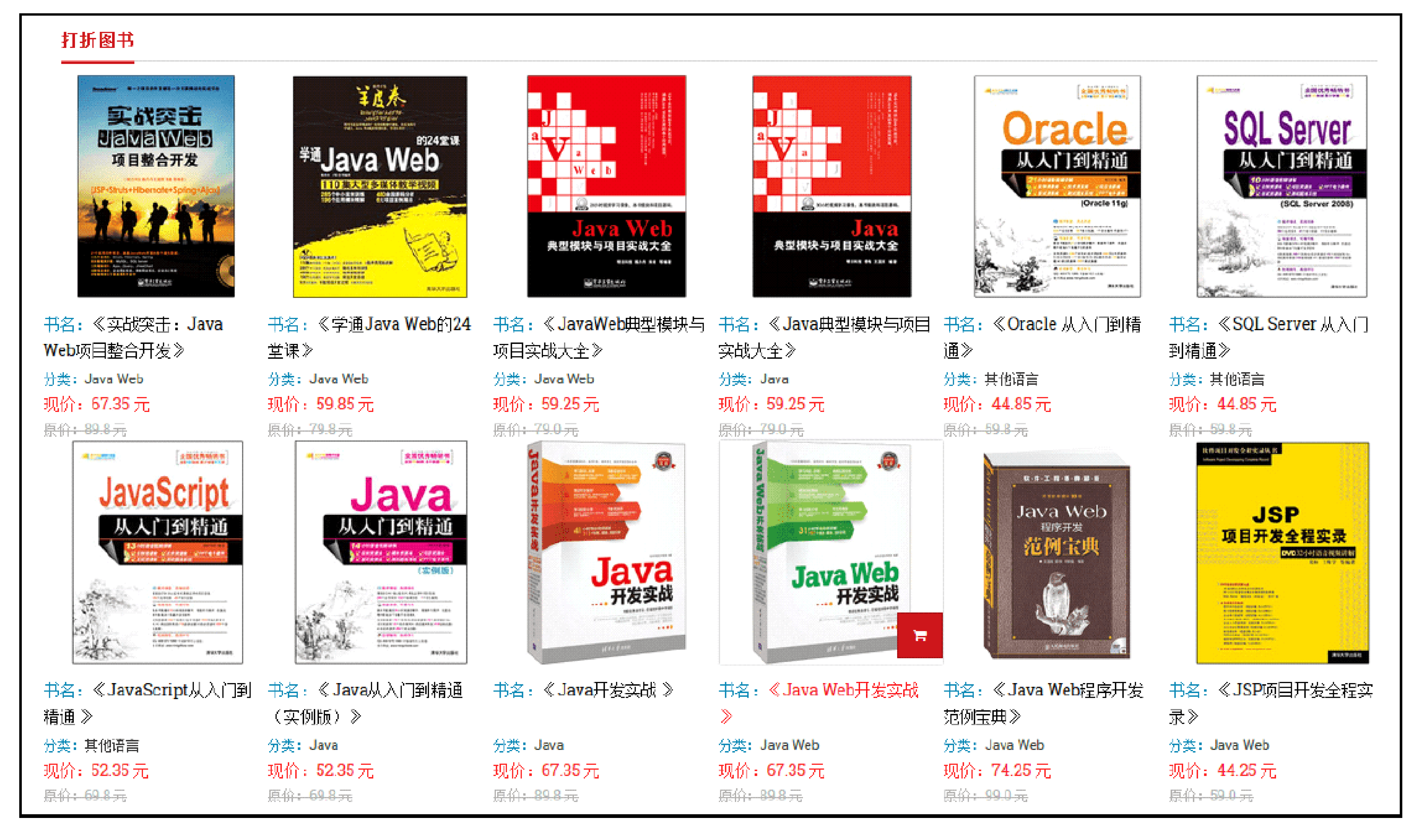
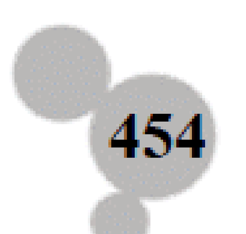


图 22.27 显示打折图书



22.8.5　实现显示热门图书功能

热门图书是指商城中点击率最高的图书，这里面获取并显示两本。在 index.jsp 文件中添加用于显示热门图书的代码，具体步骤如下。

（1）调用 ConnDB 类的 executeQuery 方法执行 SQL 语句，用于从数据表中查询点击率最高的两本图书，这里需要编写一个倒序排列的 SQL 语句。另外，还需要定义保存图书信息的变量。具体代码如下：

```
/* 热门图书信息 */
ResultSet rs_hot = conn
        .executeQuery("select top 2 ID,BookName,nowprice,picture "
        +"from tb_book order by hit desc");             //查询热门图书信息
int hot_ID = 0;                                          //保存热门图书 ID 的变量
String hot_bookName = "";                                //保存热门图书名称的变量
float hot_nowprice = 0;                                  //保存热门图书价格的变量
String hot_picture = "";                                 //保存热门图书图片的变量
```

（2）将获取到的图书信息显示到页面的热门图书展示区，具体方法同 22.8.3 节的显示最新上架图书基本相同，这里不再赘述。

运行程序，将显示图 22.28 所示的热门图书。



图 22.28　显示热门图书

22.9　购物车模块

22.9.1　购物车模块概述

在图书商城中，会员登录后，单击某图书可以进入显示图书的详细信息页面（见图 22.29），在该页面中，单击“添加到购物车”按钮即可将该图书添加到购物车，然后填写物流信息（见图 22.30），并单击“结

账”按钮，将弹出图 22.31 所示的“支付”对话框，如果已经申请到支付宝接口，并实现相应的编码，扫描对话框中的二维码即可使用支付宝进行支付（由于本项目中未提供连接支付宝接口的编码，所以无法真正支付）。最后单击“支付”按钮，生成订单并显示自动生成的订单号，如图 22.32 所示。

图 22.29　图书详细信息页面

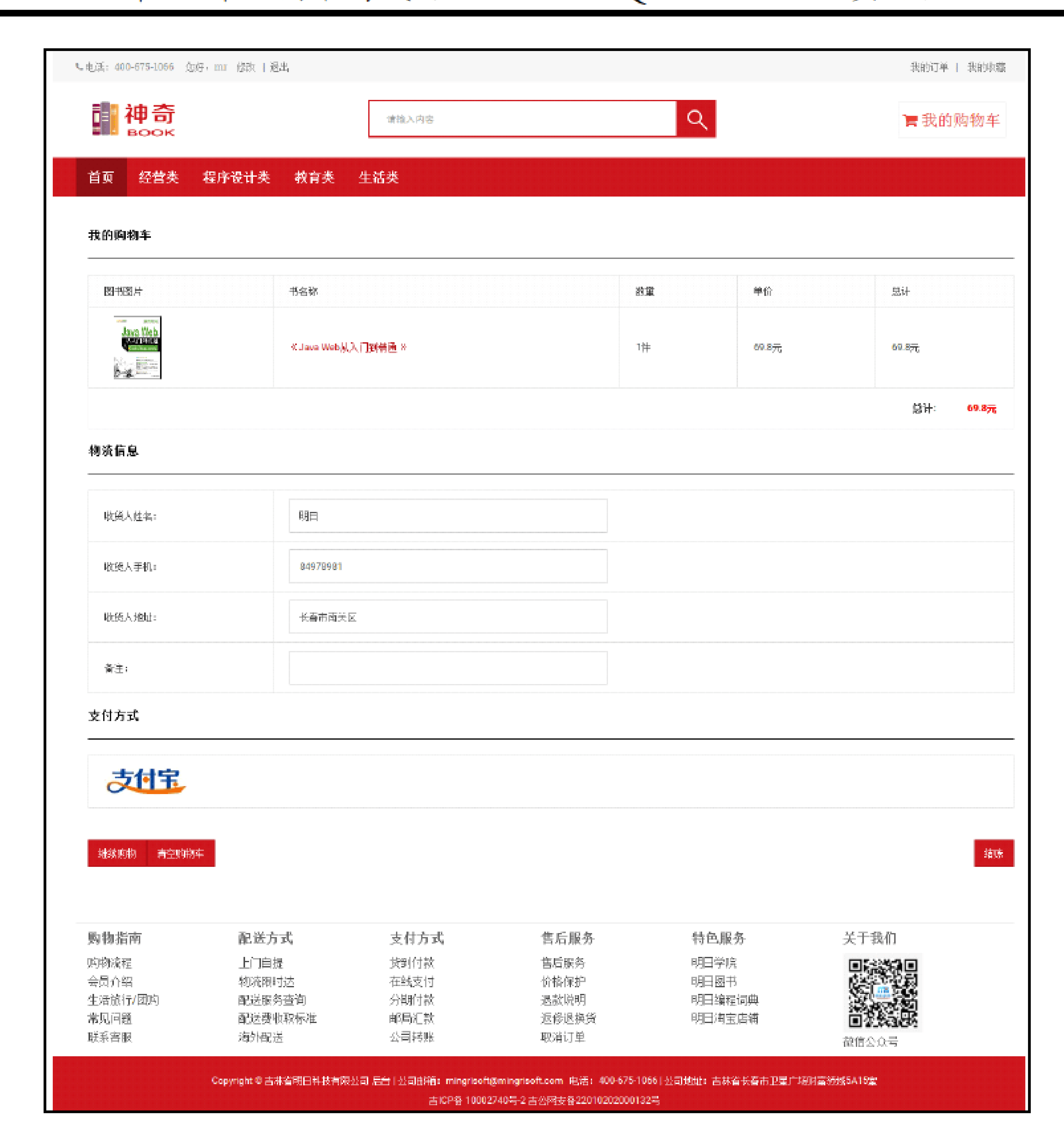


图 22.30　查看购物车页面

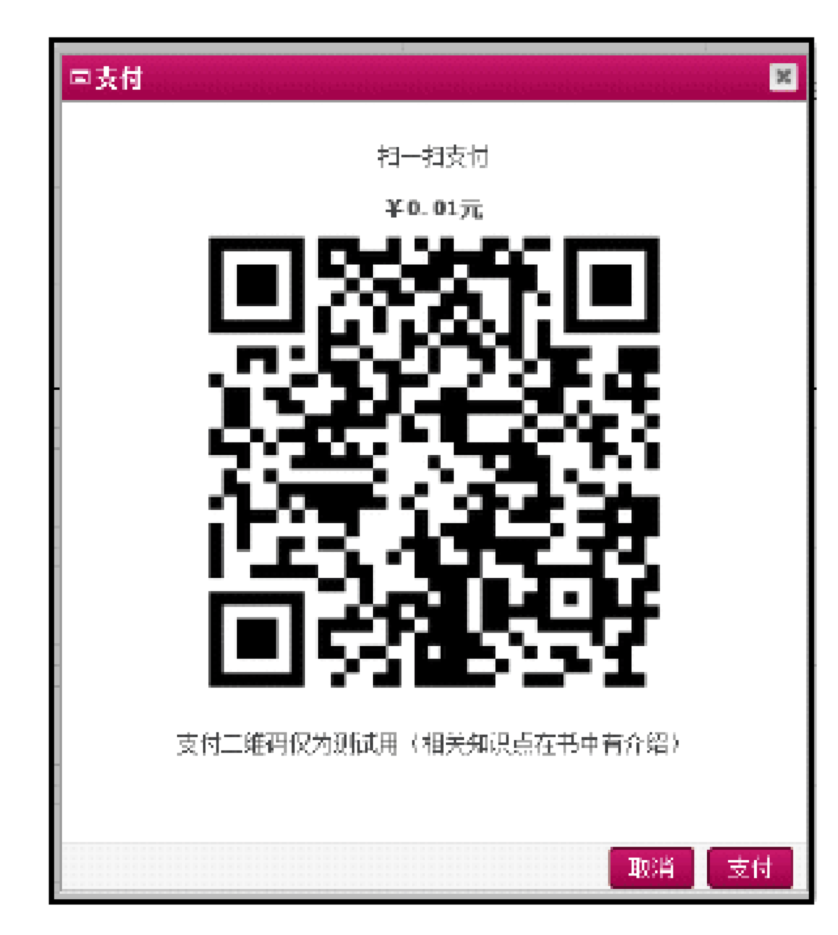


图 22.31　支付对话框

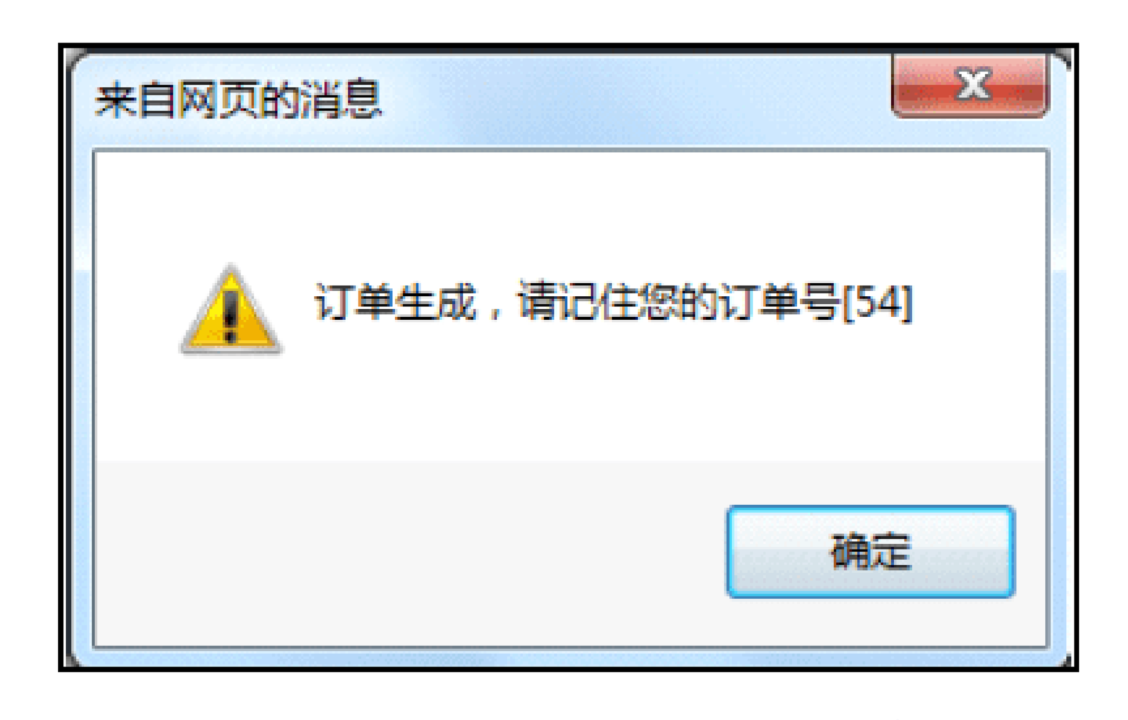


图 22.32　显示生成的订单号

22.9.2　实现显示图书详细信息功能

在首页单击任何图书名称或者图书图片时，都将显示该图书的详细信息页面。本项目中图书详细信息页面为 bookDetail.jsp。创建 bookDetail.jsp 文件的具体步骤如下。



（1）编写以下代码，用于导入 java.sql 包中的 ResultSet 类，并且创建 ConnDB 类的对象。

```
<%@ page import="java.sql.ResultSet"%>            <%-- 导入 java.sql.ResultSet 类 --%>
<%-- 创建 com.tools.ConnDB 类的对象 --%>
<jsp:useBean id="conn" scope="page" class="com.tools.ConnDB" />
```

（2）编写用于根据获取的图书 ID 查询图书信息的代码。具体的方法是：首先获取图书 ID，然后根据该图书 ID 从数据表中获取需要的图书信息，如果找到对应的图书，则将图书信息保存到相应的变量中，最后关闭数据库连接。具体代码如下：

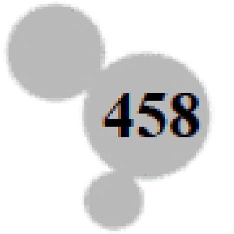
```
<%
    int typeSystem = 0;                                            //保存图书类型 ID 的变量
    int ID = Integer.parseInt(request.getParameter("ID"));         //获取图书 ID
    if (ID > 0) {
        ResultSet rs = conn.executeQuery("select ID,BookName,Introduce,nowprice,picture, "
        + " price,typeID from tb_book where ID=" + ID);            //根据 ID 查询图书信息
        String bookName = "";                                      //保存图书名称的变量
        float nowprice = (float) 0.0;                              //保存图书现价的变量
        float price = (float) 0.0;                                 //保存图书原价的变量
        String picture = "";                                       //保存图书图片的变量
        String introduce = "";                                     //保存图书描述的变量
        if (rs.next()) {                                           //如果找到对应的图书信息
            bookName = rs.getString(2);                            //获取图书名称
            introduce = rs.getString(3);                           //获取图书描述
            nowprice = rs.getFloat(4);                             //获取图书现价
            picture = rs.getString(5);                             //获取图书图片
            price = rs.getFloat(6);                                //获取图书原价
            typeSystem = rs.getInt(7);                             //获取图书类别 ID
        }
        conn.close();                                              //关闭数据库连接
%>
```

（3）在图书信息显示完毕的位置编写以下代码。用于处理获取到的图书 ID 不合法的情况。具体的方法是通过 JavaScript 弹出一个提示框，并且返回到网站的首页。

```
<%
    } else {                                                       //获取到的 ID 不合法
        out.println("<script language='javascript'>alert('您的操作有误');"
            +"window.location.href='index.jsp';</script>");
    }
%>
```

（4）在“添加到购物车”按钮的 onclick 属性中，调用自定义的 JavaScript 函数 addCart，用于验证图书数量是否合法，如果不合法则给出提示，并且返回，否则将页面转到添加到购物车页面。addCart 函数的具体代码如下

```
<script src="js/jquery.1.3.2.js" type="text/javascript"></script>
```



```
<script type="text/javascript">
    function addCart() {
        var num = $('#shuliang').val();                    //获取输入的图书数量
        //验证输入的数量是否合法
        if (num < 1) {                                     //如果输入的数量不合法
            alert('数量不能小于 1！');
            return;
        }
        //调用添加到购物车页面，实现将该图书添加到购物车
        window.location.href="cart_add.jsp?bookID=<%=ID%>&num="+num;
    }
</script>
```

说明

在上面的代码段中，cart_add.jsp 文件是用于将图书添加到购物车的处理页。后面的问号“?”，用于标识它后面的是要传递的参数，多个参数间用“&”分隔。

在已经运行的图书商城的首页中，单击某本图书的名称（如《Java Web 从入门到精通》）或者图片，都将进入图 22.33 所示的显示该图书的详细信息页面。



图 22.33　显示图书详细信息页面

22.9.3　创建购物车图书模型类 Bookelement

在 com.model 包中，创建一个名称为 Bookelement 的 Java 类。在该类中，添加 3 个公有类型的属

性，分别表示图书 ID、当前的价格和数量。Bookelement 类的具体代码如下：

```
public class Bookelement {
    public int ID;                          //定义图书 ID 变量
    public float nowprice;                  //定义现价变量
    public int number;                      //定义数量变量
}
```

22.9.4 实现添加到购物车功能

在图 22.33 中，单击“添加到购物车”按钮，即可将该图书添加到购物车。实现将图书添加到购物车的页面是 cart_add.jsp，编写该文件的具体步骤如下。

（1）在项目的 WebContent/front 节点下，创建一个名称为 cart_add.jsp 的 JSP 文件，并且在该文件中，添加以下代码。用于导入 java.sql 包中的 ResultSet 类、向量类以及图书模型类，并且创建 ConnDB 类的对象。

```
<%@ page import="java.sql.ResultSet"%>              <%-- 导入 java.sql.ResultSet 类 --%>
<%@ page import="java.util.Vector"%>                <%-- 导入 Java 的向量类 --%>
<%@ page import="com.model.Bookelement"%>           <%-- 导入购物车图书模型类 --%>
<jsp:useBean id="conn" scope="page" class="com.tools.ConnDB"/> <%-- 创建 ConnDB 类的对象 --%>
```

（2）实现添加购物车功能。首先获取会员账号和图书量，并判断是否登录，如果没有登录，则重定向到会员登录页要求登录，然后将图书基本信息保存到模型类的对象 mybookelement 中，再把该图书添加到购物车中，最后将页面跳转到查看购物车页，显示购物车内的图书。具体代码如下：

```
<%
    String username=(String)session.getAttribute("username");    //获取会员账号
    String num = (String) request.getParameter("num");            //获取图书数量
    //如果没有登录，将跳转到登录页面
    if (username == null || username == "") {
        response.sendRedirect("login.jsp");                       //重定向页面到会员登录页面
        return;                                                   //返回
    }
    int ID = Integer.parseInt(request.getParameter("bookID"));    //获取图书 ID
    String sql = "select * from tb_book where ID=" + ID;          //定义根据图书 ID 查询图书信息的 SQL 语句
    ResultSet rs = conn.executeQuery(sql);                        //根据图书 ID 查询图书
    float nowprice = 0;                                           //定义保存图书价格的变量
    if (rs.next()) {                                              //如果查询到指定图书
        nowprice = rs.getFloat("nowprice");                       //获取该图书的价格
    }
    //创建保存购物车内图书信息的模型类的对象 mybookelement
    Bookelement mybookelement = new Bookelement();
    mybookelement.ID = ID;                                        //将图书 ID 保存到 mybookelement 对象中
    mybookelement.nowprice = nowprice;                            //将图书价格保存到 mybookelement 对象中
```

```
    mybookelement.number = Integer.parseInt(num);           //将购买数量保存到 mybookelement 对象中
    boolean Flag = true;                                    //记录购物车内是否已经存在所要添加的图书
    Vector cart = (Vector) session.getAttribute("cart");    //获取购物车对象
    if (cart == null) {                                     //如果购物车对象为空
        cart = new Vector();                                //创建一个购物车对象
    } else {
        //判断购物车内是否已经存在所购买的图书
        for (int i = 0; i < cart.size(); i++) {
            Bookelement bookitem = (Bookelement) cart.elementAt(i);//获取购物车内的一本图书
            if (bookitem.ID == mybookelement.ID) {          //如果当前要添加的图书已经在购物车中
                //直接改变购物数量
                bookitem.number = bookitem.number + mybookelement.number;
                cart.setElementAt(bookitem, i);             //重新保存到购物车中
                Flag = false;                   //设置标记变量 Flag 为 false，代表购物车中存在该图书
            }
        }
    }
    if (Flag)                                               //如果购物车内不存在该图书
        cart.addElement(mybookelement);                     //将要购买的图书保存到购物车中
    session.setAttribute("cart", cart);                     //将购物车对象添加到 Session 中
    conn.close();                                           //关闭数据库的连接
    response.sendRedirect("cart_see.jsp");                  //重定向页面到查看购物车页面
%>
```

说明

由于添加到购物车页面是一个处理页，主要是把一些信息进行保存的，所以没有呈现结果。

22.9.5　实现查看购物车功能

在将图书添加到购物车后，需要把页面跳转到查看购物车页面，用于显示已经添加到购物车中的图书。查看购物车页面为 cart_see.jsp。该文件的具体实现步骤如下。

（1）在项目的 WebContent/front 节点下，创建一个名称为 cart_see.jsp 的 JSP 文件，添加下面的代码。用于判断是否登录，如果没有登录，则进入登录页面进行登录，否则获取购物车对象，并且根据获取结果进行显示。

```
<%
    String username = (String) session.getAttribute("username");        //获取会员账号
    //如果没有登录，将跳转到登录页面
    if (username == "" || username == null) {
        response.sendRedirect("login.jsp");                             //重定向页面到会员登录页面
        return;                                                         //返回
    } else {
        Vector cart = (Vector) session.getAttribute("cart");            //获取购物车对象
```

```
            if (cart == null || cart.size() == 0) {                    //如果购物车为空
                response.sendRedirect("cart_null.jsp");                //重定向页面到购物车为空页面
            } else {
%>
```

（2）滚动到页面的最底部，添加以下代码，用于结束步骤（1）中的两个 if 语句。

```
<%          }
     } %>
```

（3）遍历购物车中的图书，并获取要显示的信息。具体的实现方法是：首先根据购物车中保存的图书 ID，从图书信息表中获取其详细信息（主要是图书名称和图书封面图片），并保存到相应的变量中，最后不要忘记关闭数据库连接。具体代码如下：

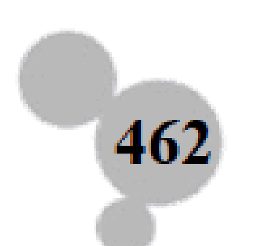
```
<%
     float sum = 0;
     DecimalFormat fnum = new DecimalFormat("##0.0");                  //定义显示金额的格式
     int ID = -1;                                                      //保存图书 ID 的变量
     String bookname = "";                                             //保存图书名称的变量
     String picture = "";                                              //保存图书图片的变量
     //遍历购物车中的图书
     for (int i = 0; i < cart.size(); i++) {
          Bookelement bookitem = (Bookelement) cart.elementAt(i);     //获取一个图书
          sum = sum + bookitem.number * bookitem.nowprice;            //计算总计金额
          ID = bookitem.ID;                                            //获取图书 ID
          if (ID > 0) {
               ResultSet rs_book = conn.executeQuery("select * from tb_book where ID=" + ID);
               if (rs_book.next()) {
                    bookname = rs_book.getString("bookname");          //获取图书名称
                    picture = rs_book.getString("picture");            //获取图书封面图片
               }
               conn.close();                                           //关闭数据库连接
     }
%>
```

（4）在购物车信息显示完毕的位置插入以下代码，用于结束步骤（3）中的 for 循环，以及格式化总计金额。格式化后的格式为“##0.0”，即小数点后保留一位小数。

```
<%
     }
     String sumString = fnum.format(sum);                              //格式化总计金额
%>
```

22.9.6 实现调用支付宝完成支付功能

在查看购物车页面中，单击“结账”按钮，首先会弹出“支付”对话框，在该对话框中，扫描二



维码将调用支付宝完成支付功能。在编写本书时，实现在网站中加入支付功能的基本方法如下。

1．注册支付宝企业账户

进入支付宝开发平台（蚂蚁金服开放平台）。单击“注册”超链接，进入“注册－支付宝”页面，在该页面中选择“企业账户”选项卡，然后按照向导进行操作即可。

2．完成支付宝实名认证

注册支付宝企业账户后，会要求进行实名认证。准备以下资料后，单击“企业实名信息填写”按钮，按照向导完成实名认证。

☑ 营业执照影印件。

☑ 对公银行账户，可以是基本户或一般户。

☑ 法宝代表人的身份证影印件。

说明

如果您是代理人，除以上资料外，还需要准备您的身份证影印件和企业委托书，必须盖有公司公章或者账务专用章。

3．申请支付套餐

支付宝提供了多种支付套餐。一般情况下，我们可以选择“即时到账”套餐。该套餐可以让用户在线向开发者的支付宝账号支付资金，并且交易资金即时到账。要申请“即时到账”套餐，可以直接在浏览器的地址栏中输入 URL 地址 https://b.alipay.com/order/productDetail.htm?productId=2015110218012942，在进入的页面中，直接单击“在线申请”按钮，然后按照向导进行操作。

申请好套餐后，会有一个审核阶段，审核通过才能使用该接口。通常情况下，2~5 天会有申请结果。

4．生成与配置密钥

进行开发时，需要提供商户的私钥和支付宝的公钥，这些内容也可以到支付宝开发平台中获取，对应的 URL 地址为 https://doc.open.alipay.com/docs/doc.htm?spm=a219a.7629140.0.0.ulCjKD&treeId=193&articleId=105310&docType=1。在该页面根据提示进行操作即可。

5．下载 Demo

前面的工作准备就绪后，就可以开发测试支付功能了。这时，可以下载支付宝开发平台提供的即时到账交易接口的 Demo，然后根据 Demo 中的说明进行开发测试即可。

22.9.7　实现保存订单功能

单击“结账”按钮，即可保存该订单。保存订单页面为 cart_order.jsp，在该页面中实现保存订单功能。首先判断购物车是否为空，不为空时，再判断会员账户是否合法，只有会员账户合法时，才保存订单。在保存订单信息时，需要分别向订单主表和订单明细表插入数据。具体代码如下：

```
<%
    if (session.getAttribute("cart") == "") {                          //判断购物车对象是否为空
        out.println(
                "<script language='javascript'>alert('您还没有购物!');"
                +"window.location.href='index.jsp';</script>");
    }
    String Username = (String) session.getAttribute("username");       //获取输入的账户名称
    if (Username != "") {
        try {                                                           //捕捉异常
            ResultSet rs_user = conn.executeQuery("select * from tb_Member where username='"
            + Username + "'");
            if (!rs_user.next()) {          //如果获取的账户名称在会员信息表中不存在（表示非法会员）
                session.invalidate();  //销毁 Session
                out.println(
                        "<script language='javascript'>alert('请先登录后，再进行购物!'); "
                        +"window.location.href='index.jsp';</script>");
                return;                                                 //返回
            } else {                                                    //如果合法会员，则保存订单
                //获取输入的收货人姓名
                String recevieName = chStr.chStr(request.getParameter("recevieName"));
                String address = chStr.chStr(request.getParameter("address"));   //获取输入的收货人地址
                String tel = request.getParameter("tel");               //获取输入的电话号码
                String bz = chStr.chStr(request.getParameter("bz"));//获取输入的备注
                int orderID = 0;                                        //定义保存订单 ID 的变量
                Vector cart = (Vector) session.getAttribute("cart");    //获取购物车对象
                int number = 0;                                         //定义保存图书数量的变量
                float nowprice = (float) 0.0;                           //定义保存图书价格的变量
                float sum = (float) 0;                                  //定义图书金额的变量
                float Totalsum = (float) 0;                             //定义图书件数的变量
                boolean flag = true;                                    //标记订单是否有效，为 true 表示有效
                int temp = 0;                                           //保存返回自动生成的订单号的变量
                int ID = -1;
                //插入订单主表数据
                float bnumber = cart.size();
                String sql = "insert into tb_Order(bnumber,username, recevieName,address, "
                        +"tel,bz) values("+ bnumber + ",'" + Username + "','" + recevieName
                        + "','" + address + "','" + tel+ "','" + bz + "')";
                temp = conn.executeUpdate_id(sql);                      //保存订单主表数据
                if (temp == 0) {                                        //如果返回的订单号为 0，表示不合法
                    flag = false;
                } else {
                    orderID = temp;                                     //把生成的订单号赋值给订单 ID 变量
                }
                String str = "";                                        //保存插入订单详细信息的 SQL 语句
                for (int i = 0; i < cart.size(); i++) {                 //插入订单明细表数据
                    //获取购物车中的一个图书
```

```
                    Bookelement mybookelement = (Bookelement) cart.elementAt(i);
                    ID = mybookelement.ID;                              //获取图书 ID
                    nowprice = mybookelement.nowprice;                  //获取图书价格
                    number = mybookelement.number;                      //获取图书数量
                    sum = nowprice * number;                            //计算图书金额
                    str = "insert into tb_order_Detail (orderID,bookID,price,number)"
                            +" values(" + orderID + ","+ ID + "," + nowprice + ","
                            + number + ")";                             //插入订单明细的 SQL 语句
                    temp = conn.executeUpdate(str);                     //保存订单明细
                    Totalsum = Totalsum + sum;                          //累加合计金额
                    if (temp == 0) {                                    //如果返回值为 0，表示不合法
                        flag = false;
                    }
                }
                if (!flag) {                                            //如果订单无效
                    out.println("<script language='javascript'>alert('订单无效');"
                                +"history.back();</script>");
                } else {
                    session.removeAttribute("cart");                    //清空购物车
                    out.println("<script language='javascript'>alert('订单生成，请记住您"
                            +"的订单号[" + orderID
                    + "]');window.location.href='index.jsp';</script>");    //显示生成的订单号
                }
                conn.close();                                           //关闭数据库连接
            }
        } catch (Exception e) {                                         //处理异常
            out.println(e.toString());                                  //输出异常信息
        }
    } else {
        session.invalidate();                                           //销毁 Session
        out.println(
                "<script language='javascript'>alert('请先登录后，再进行购物!');"
                +"window.location.href='index.jsp';</script>");
    }
%>
```

22.10　小　结

本章主要通过 JavaWeb+SQL Server 技术讲解了一个图书商城的实现过程，通过本章的学习，读者应该熟练掌握使用 JDBC 操作数据库技术，并且熟悉使用支付宝进行在线支付的实现流程。

第 23 章　房屋中介管理系统（C# +SQL Server 2014 实现）

随着信息技术的日益发展，作为房屋中介公司的工作人员，希望通过使用计算机来代替烦琐和大量的手工操作，以便达到事半功倍的效果，这样能够使房屋中介对求租人信息、出租人信息和房源信息的管理实现系统化、规范化及自动化。本章将通过使用 C#+SQL Server 2014 技术开发一个房屋中介管理系统。通过本章的学习，可以掌握以下要点：

☑　数据的定位查询和模糊查询
☑　进行严格的数据检验
☑　图形化显示房源信息
☑　使用存储过程
☑　实现断开式数据库连接
☑　使用图标显示房屋状态

23.1　开 发 背 景

房屋中介管理系统是房屋中介机构不可缺少的一部分，能够为操作人员和用户提供充足的信息和快速查询手段。但一直以来人们使用传统人工的方式管理房屋出租、求租等房屋信息，这种管理存在着许多缺点，如效率低、保密性差等，时间一长，将产生大量的文件和数据，这样给查找、更新和维护房屋信息带来了不少的困难。而房屋中介管理系统的出现改变了这一现状，它是一款非常实用的房屋中介管理软件，使用该软件，不仅可以详细地记录房源信息和用户信息等，同时还能够自动查找和客户需求相匹配的房源，在方便客户的同时又提高了使用者的工作质量和效率。

23.2　需 求 分 析

通过与某房屋中介公司的沟通和需求分析，要求系统具有以下功能：

☑　由于操作人员的计算机知识有限，因此要求系统具有良好的人机界面；
☑　如果系统的使用对象较多，则要求有较好的权限管理；
☑　方便的数据查询，支持自定义条件查询；

☑　自动匹配房源和求房意向信息；
☑　使用垃圾信息处理机制释放空间；
☑　在相应的权限下，可方便地删除数据；
☑　数据计算自动完成，尽量减少人工干预。

23.3　系统设计

23.3.1　系统目标

本系统属于小型的数据库系统，可以对房源和租赁人等进行有效的管理。通过本系统可以达到以下目标：

☑　系统采用人机交互方式，界面美观友好，信息查询灵活、方便，数据存储安全可靠；
☑　灵活的批量录入数据，使信息传递更快捷；
☑　实现垃圾信息清理；
☑　实现后台监控功能；
☑　实现各种查询，如定位查询、模糊查询等；
☑　实现图形化显示房源信息；
☑　对用户输入的数据，进行严格的数据检验，尽可能避免人为错误；
☑　系统最大限度地实现了易安装性、易维护性和易操作性。

23.3.2　系统功能结构

房屋中介管理系统的部分功能结构如图 23.1 所示。

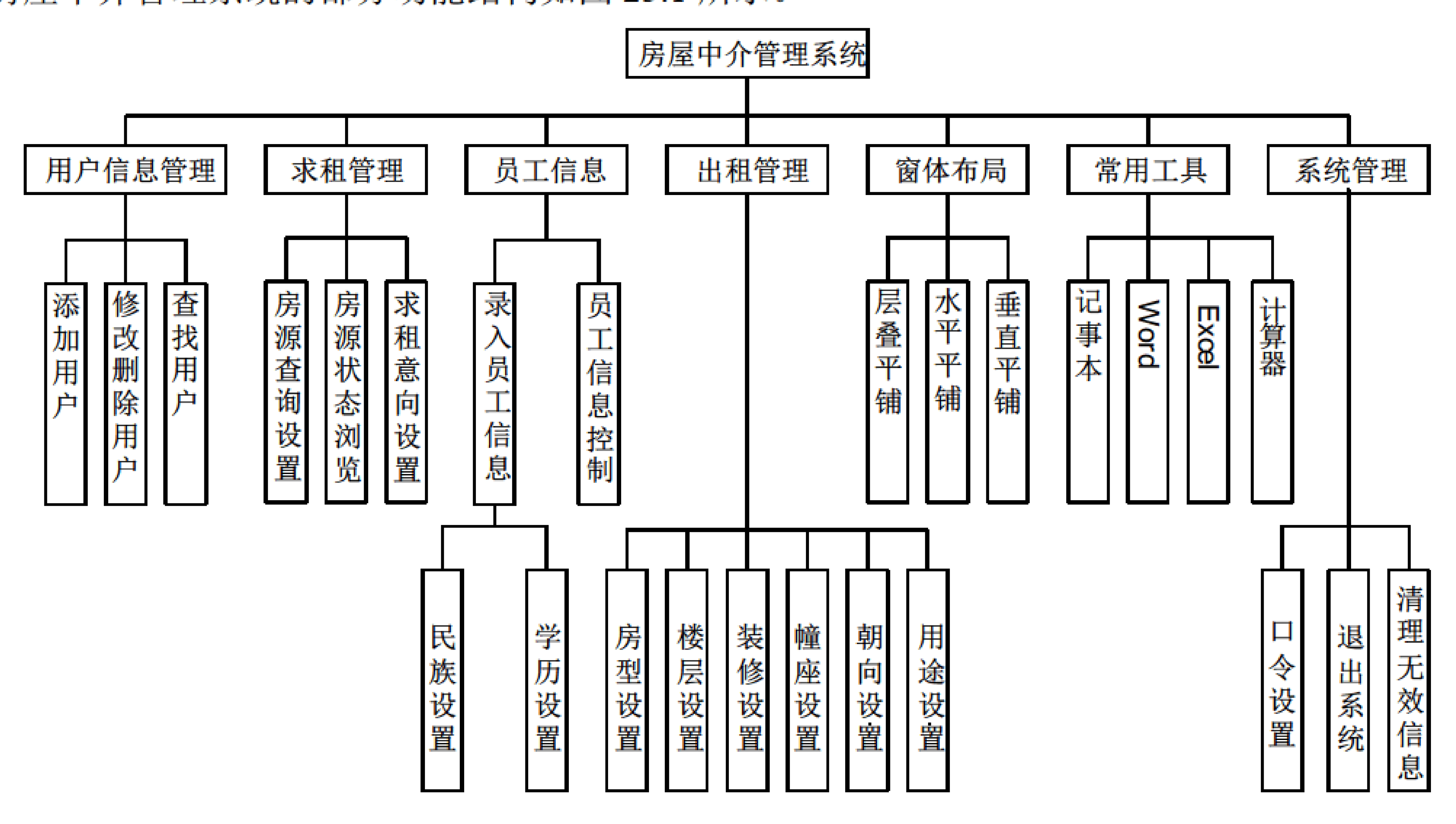


图 23.1　房屋中介管理系统的部分功能结构

23.3.3　业务流程图

房屋中介管理系统的业务流程图如图 23.2 所示。

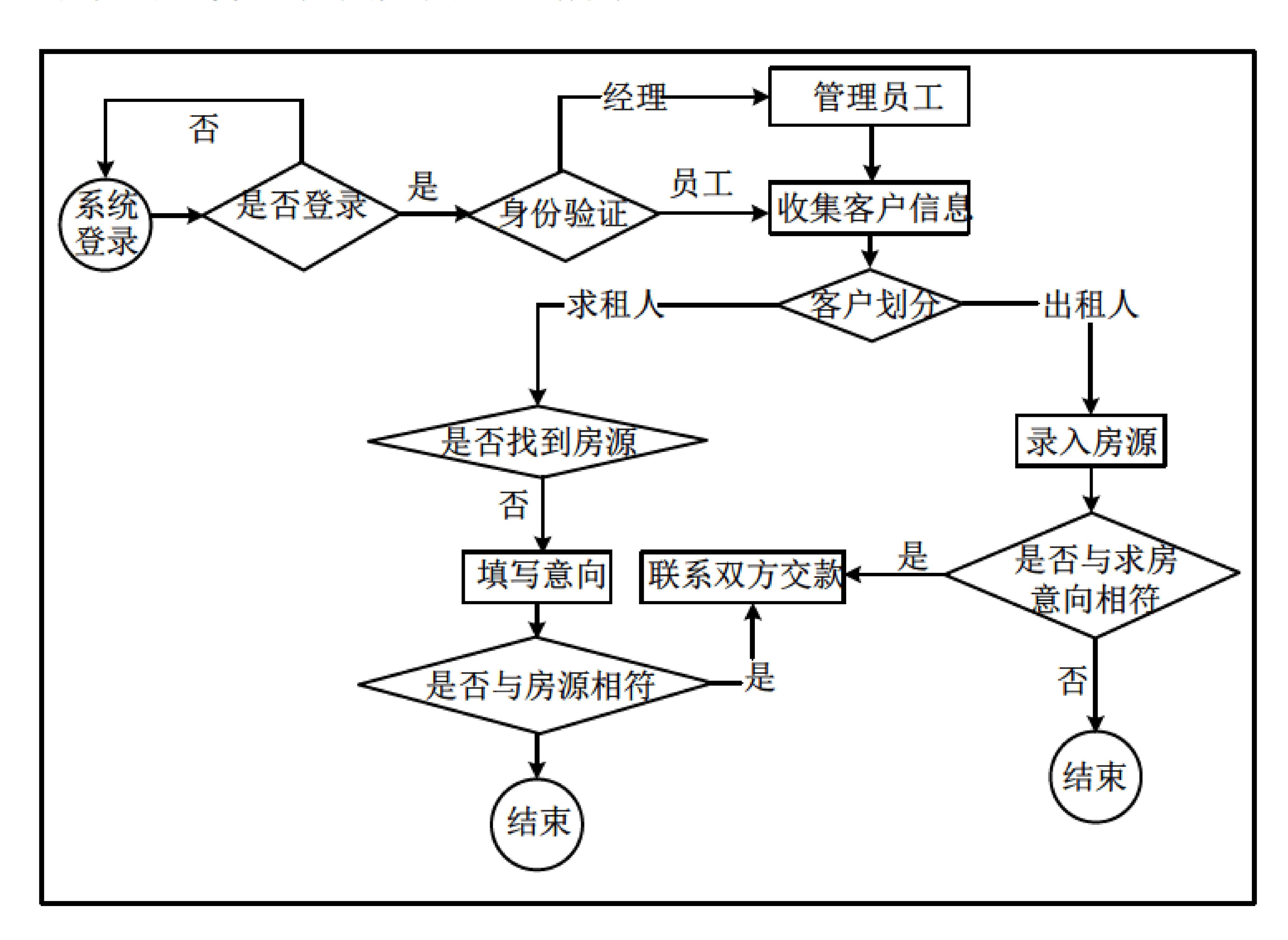


图 23.2　房屋中介管理系统的业务流程图

23.3.4　业务逻辑编码规则

遵守程序编码规则所开发的程序，代码清晰、整洁、方便阅读，并可以提高程序的可读性，真正做到“见其名知其意”。本节从数据库设计和程序编码两个方面介绍程序开发中的编码规则。

1．数据库对象命名规则

☑　数据库命名规则

数据库命名以字母 db 开头（小写），后面加数据库相关英文单词或缩写。下面将举例说明，如表 23.1 所示。

表 23.1　数据库命名

数据库名称	描　　述
db_House	房屋中介管理系统数据库

☑　数据表命名规则

数据表命名以字母 tb 开头（小写），后面加数据库相关英文单词或缩写和数据表名，多个单词间用“_”分隔。下面将举例说明，如表 23.2 所示。

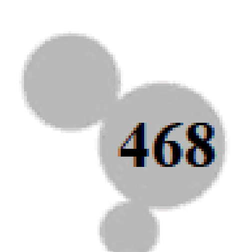


表 23.2　数据表命名

数据表名称	描　　述
tb_employee	员工信息表
tb_MoneyAndInfo	收费信息表

☑　字段命名规则

字段一律采用英文单词或词组（可利用翻译软件）命名，如找不到专业的英文单词或词组，可以用相同意义的英文单词或词组代替，另外，单词或单词缩写之间可以使用“_”分隔。下面将举例说明，表 23.3 所示为库存信息表中的部分字段。

表 23.3　字段命名

字 段 名 称	描　　述
employee_ID	员工编号
employee_name	员工名称
employee_sex	员工性别

2．业务编码规则

☑　员工编号

员工编号是房屋中介管理系统中员工的唯一标识，不同的员工可以通过该编号来区分（即使员工名称相同）。在本系统中该编号的命名规则：以字符串 emp 为编号前缀，加上 4 位数字作为编号的后缀，这 4 位数字从 1001 开始。例如，emp1001。

☑　客户编号

客户编号是房屋中介管理系统中客户的唯一标识，对于中介机构，它的客户分为出租人和求租人两类，不同的客户可以通过该编号来区分（即使客户名称相同）。在本系统中该编号的命名规则：以字符串 want（标识求租人）或 lend（标识出租人）为编号前缀，加上 4 位数字作为编号的后缀，这 4 位数字从 1001 开始。例如，lend1006 或 want1005。

☑　房屋编号

房屋编号是房屋中介管理系统中房源的唯一标识，它用于唯一标识某一套具体的出租房屋。在本系统中该编号的命名规则：以字符串 hou 为编号前缀，加上 4 位数字作为编号的后缀，这 4 位数字从 1001 开始。例如，hou1001。

23.3.5　程序运行环境

本系统的程序运行环境具体如下。

☑　系统开发平台：Microsoft Visual Studio 2017。

☑　系统开发语言：C#。

☑　数据库管理系统软件：SQL Server 2014。

☑　运行平台：Windows 7（SP1）/ Windows 8/Windows 10。

☑ 运行环境：Microsoft.NET Framework SDK v4.7。

23.3.6 系统预览

房屋中介管理系统由多个窗体组成，下面仅列出几个典型窗体，其他窗体参见资源包中的源程序。

主窗体如图 23.3 所示，主要实现快速链接系统的所有功能，该窗体提供两种打开子窗体的菜单：既可以通过最上面的常规菜单打开系统中的所有子窗体；也可以通过窗体左面的树型菜单来打开系统中的所有子窗体。



图 23.3 主窗体

求租人员信息窗体如图 23.4 所示，主要实现登记求租人信息，注意“手机号码”和“身份证号码”必须输入，以备后面的操作之用。出租人员信息设置窗体如图 23.5 所示，主要是完成出租人信息登记和所要出租的房屋登记。这两个窗体使用同一个类文件，即 frmPeopleInfo.cs 文件，程序根据打开的命令不同，显示或隐藏“录入房源”按钮。

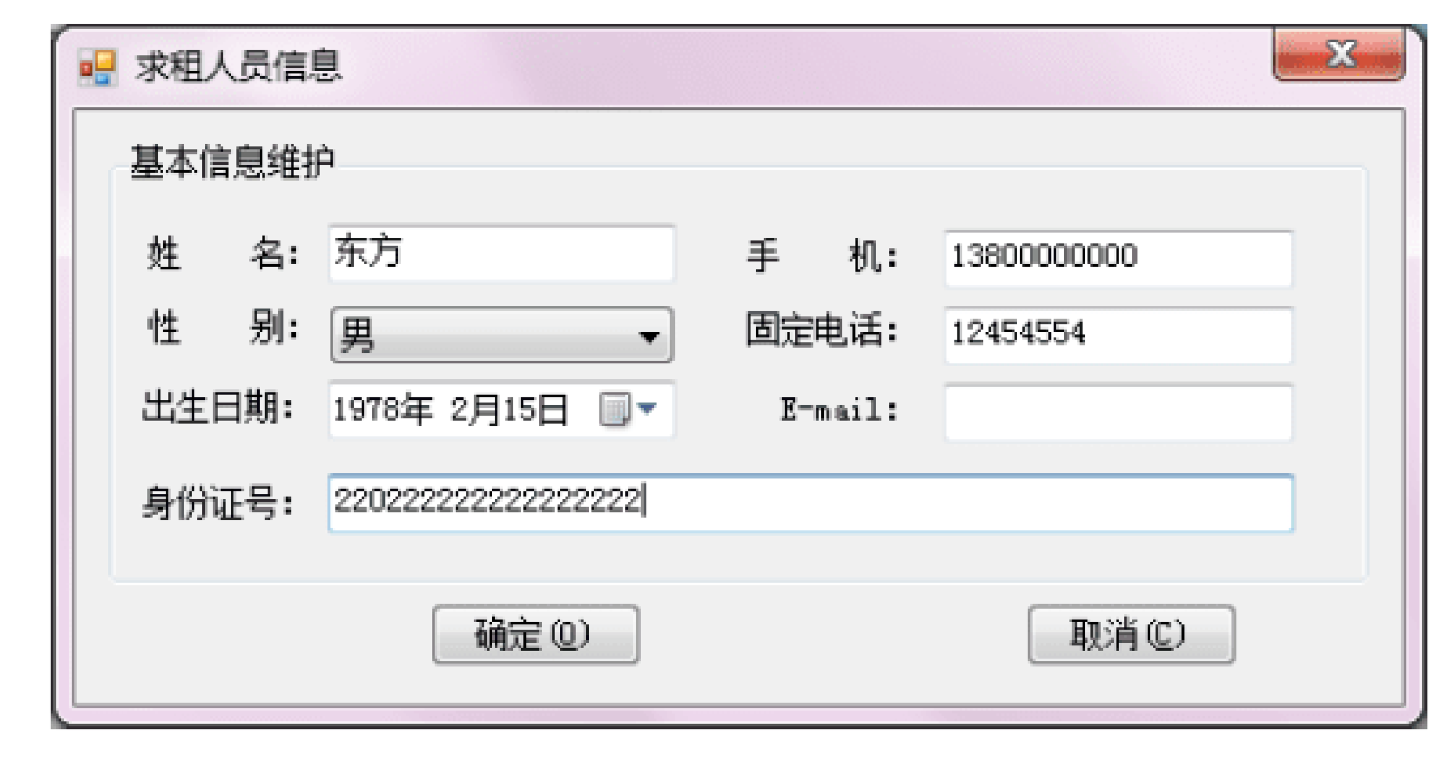



图 23.4 求租人员信息窗体

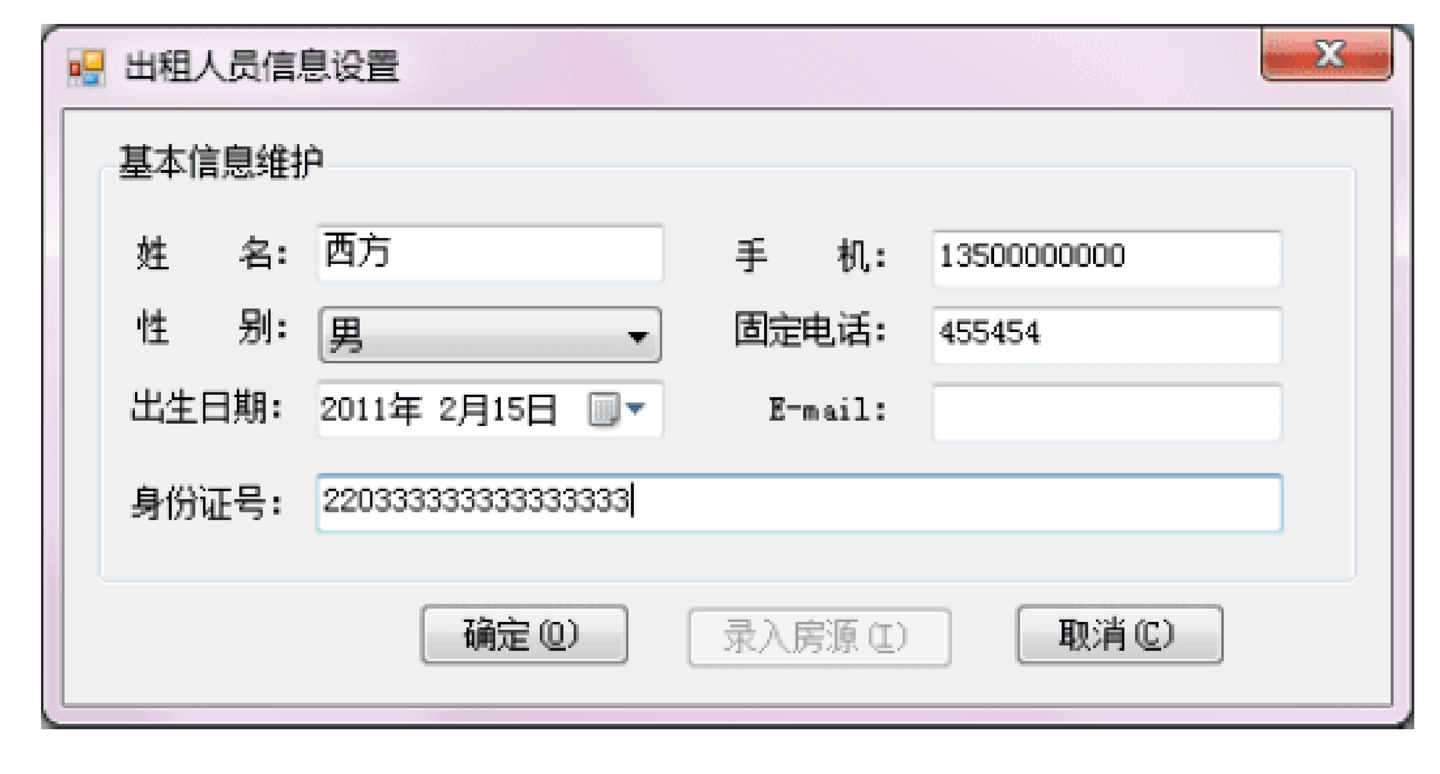


图 23.5 出租人员信息设置窗体

房屋状态查询窗体如图 23.6 所示，主要实现查询房屋的状态，房屋的状态包括已租、未租和预订 3 种状态。另外，还可以通过手机号进行预订房屋和取消预订两种操作。





图 23.6　房屋状态查询窗体

23.4　数据库设计

23.4.1　数据库概要说明

本系统采用 SQL Server 2014 作为后台数据库，数据库名称为 db_House，其中包含 15 张数据表，详细情况如图 23.7 所示。

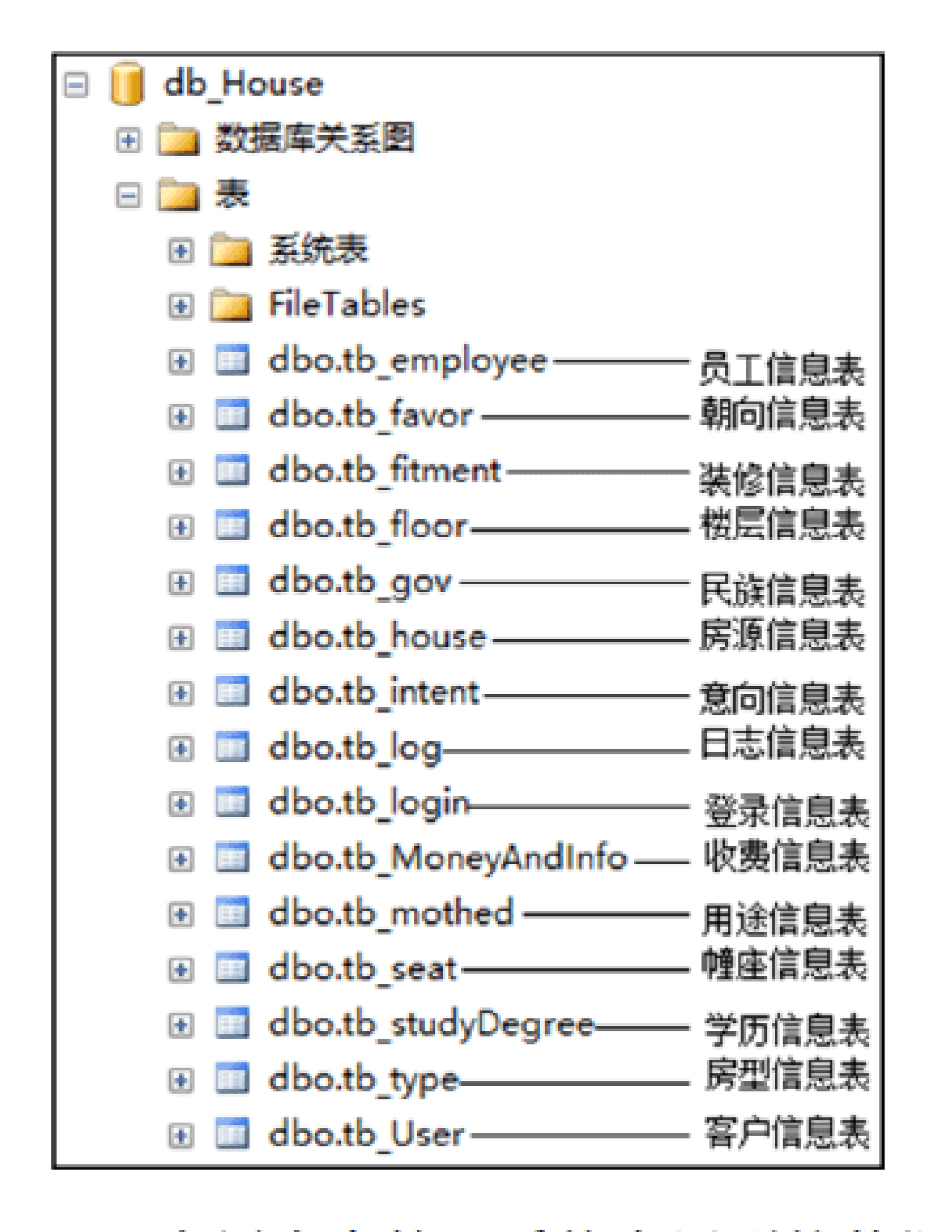
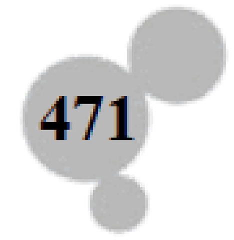


图 23.7　房屋中介管理系统中用到的数据表

23.4.2　数据库概念设计

本系统中规划出的实体主要有员工信息实体、客户信息实体、房源信息实体和意向信息实体等。员工信息实体 E-R 图如图 23.8 所示。客户信息实体 E-R 图如图 23.9 所示。



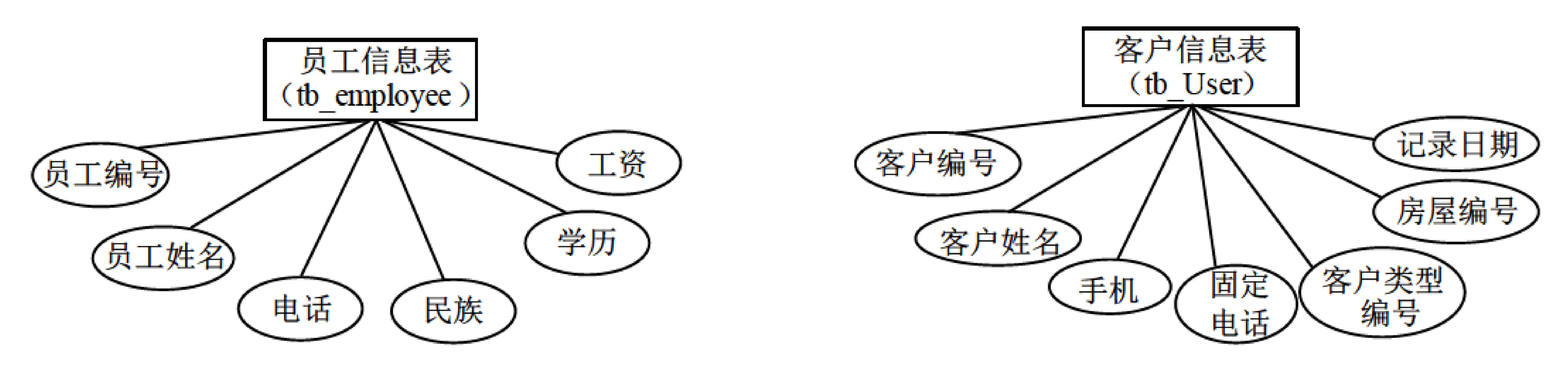


图 23.8　员工信息实体 E-R 图　　　　图 23.9　客户信息实体 E-R 图

房源信息实体 E-R 图如图 23.10 所示。意向信息实体 E-R 图如图 23.11 所示。

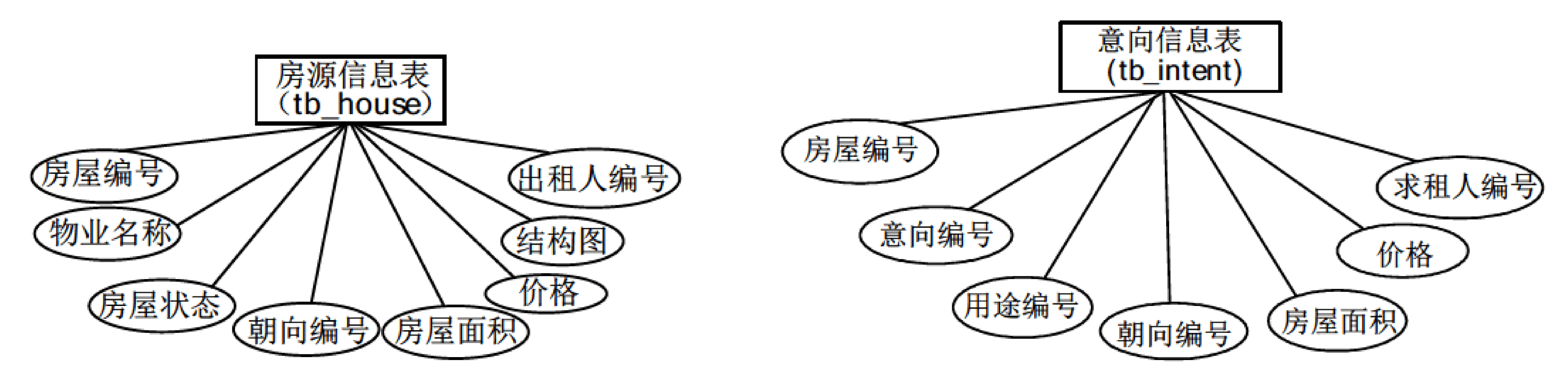


图 23.10　房源信息实体 E-RE-R 图　　　　图 23.11　意向信息实体 E-R 图

收费信息实体 E-R 图如图 23.12 所示。朝向信息实体 E-R 图如图 23.13 所示。

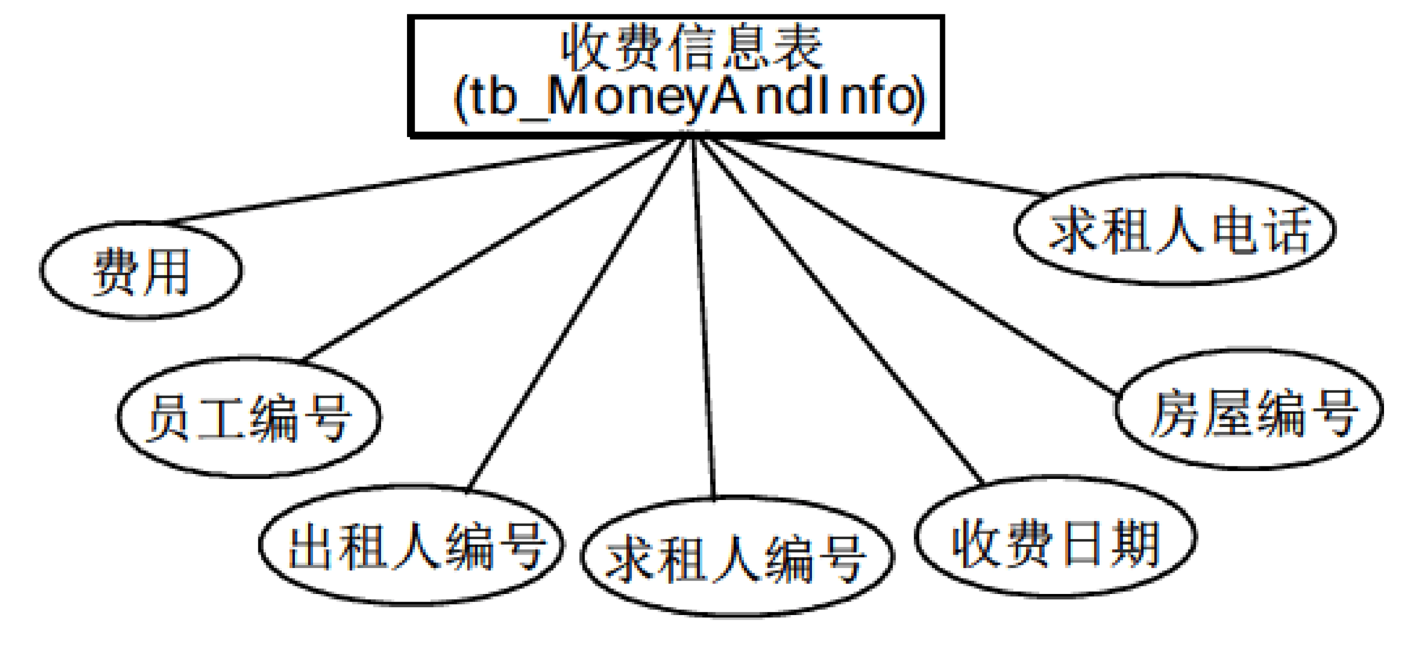


图 23.12　收费信息实体 E-R 图

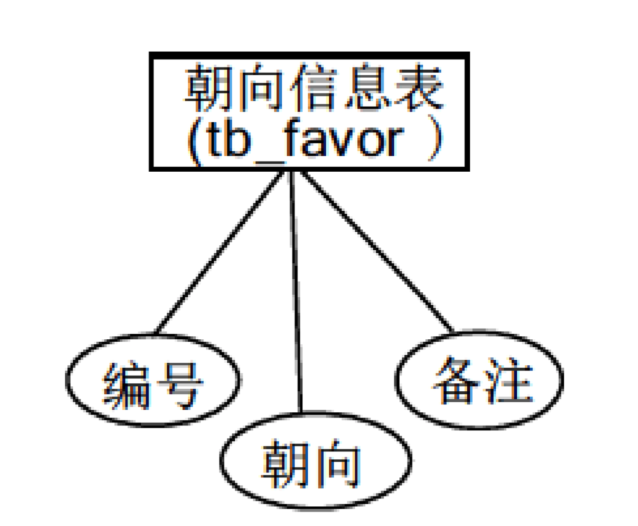


图 23.13　朝向信息实体 E-R 图

23.4.3　数据库逻辑设计

由于篇幅所限，下面对比较重要的数据表的结构进行介绍。

☑　员工信息表

员工信息表（tb_employee）用于保存员工的基本信息，该表的结构如表 23.4 所示。

表 23.4　员工信息表结构

字 段 名 称	数 据 类 型	字 段 大 小	说　　明
employee_ID	varchar	10	员工编号
employee_name	varchar	20	姓名
employee_sex	varchar	10	性别
employee_birthday	datetime	8	出生日期

续表

字 段 名 称	数 据 类 型	字 段 大 小	说　　明
employee_phone	varchar	20	电话
employee_cardID	varchar	20	身份证号
employee_address	varchar	50	地址
gov_ID	varchar	10	民族
employee_study	varchar	10	学历
employee_basepay	money	8	工资

☑　客户信息表

客户信息表（tb_User）用于保存客户信息，该表的结构如表 23.5 所示。

表 23.5　客户信息表结构

字 段 名 称	数 据 类 型	字 段 大 小	说　　明
User_IDS	varchar	10	客户编号
User_nameS	varchar	20	姓名
User_sex	varchar	4	性别
User_birth	datetime	8	出生日期
User_phone	varchar	20	手机
User_homePhone	varchar	20	宅电
User_email	varchar	30	邮箱
User_cardID	varchar	20	身份证
User_type	varchar	10	客户类型
house_ID	varchar	10	房屋编号
User_recordDate	datetime	8	记录日期

☑　房源信息表

房源信息表（tb_house）用于保存房源信息，该表的结构如表 23.6 所示。

表 23.6　房源信息表结构

字 段 名 称	数 据 类 型	字 段 大 小	说　　明
house_ID	varchar	10	房屋编号
house_companyName	varchar	50	物业名称
huose_typeID	varchar	10	房型编号
house_seatID	varchar	10	幢/座编号
house_state	varchar	10	状态
house_fitmentID	Varchar	10	装修编号
house_favorID	varchar	10	朝向编号
house_mothedID	varchar	10	用途编号
huose_map	varchar	50	结构图

续表

字 段 名 称	数 据 类 型	字 段 大 小	说　　明
house_price	money	8	价格
house_floorID	varchar	10	楼层编号
house_buildYear	varchar	10	建筑年限
house_area	varchar	20	建筑面积
house_remark	varchar	50	备注
User_IDS	varchar	10	用户编号

☑　意向信息表

意向信息表（tb_intent）用于保存求租人对房源的要求信息，该表的结构如表 23.7 所示。

表 23.7　意向信息表结构

字 段 名 称	数 据 类 型	字 段 大 小	说　　明
intent_ID	varchar	10	意向编号
User_ID	varchar	10	用户编号
huose_typeID	varchar	10	房型编号
house_seatID	varchar	10	幢/座编号
house_fitmentID	varchar	10	装修编号
house_floorID	varchar	10	楼层编号
house_favorID	varchar	10	朝向编号
house_mothedID	varchar	10	用途编号
house_price	nvarchar	8	价格
house_area	varchar	20	面积

23.5　公共类设计

在开发项目中以类的形式来组织、封装一些常用的方法和事件，不仅可以提高代码的重用率，也大大方便了代码的管理。本系统中创建了公共类 ClsCon.cs，并且还为每个数据表建立了自己的实体类和方法类。在此只介绍一张数据表所对应的实体类和方法类，其他数据表所对应的类，可参见本书附带资源包中的源程序。

23.5.1　程序文件架构

部分主文件架构如图 23.14 所示。



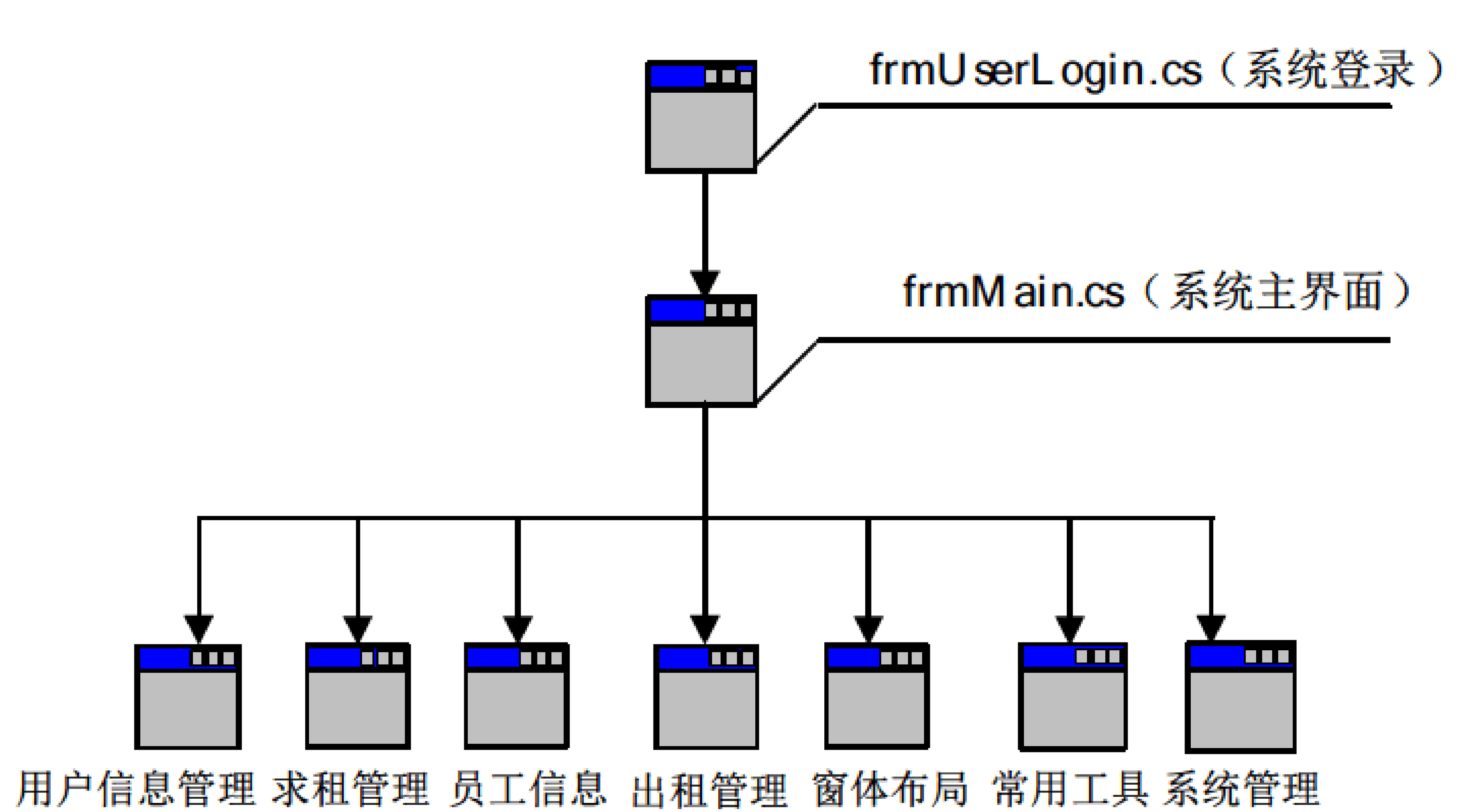


图 23.14　部分主文件架构

员工信息和用户信息管理文件架构如图 23.15 和图 23.16 所示。

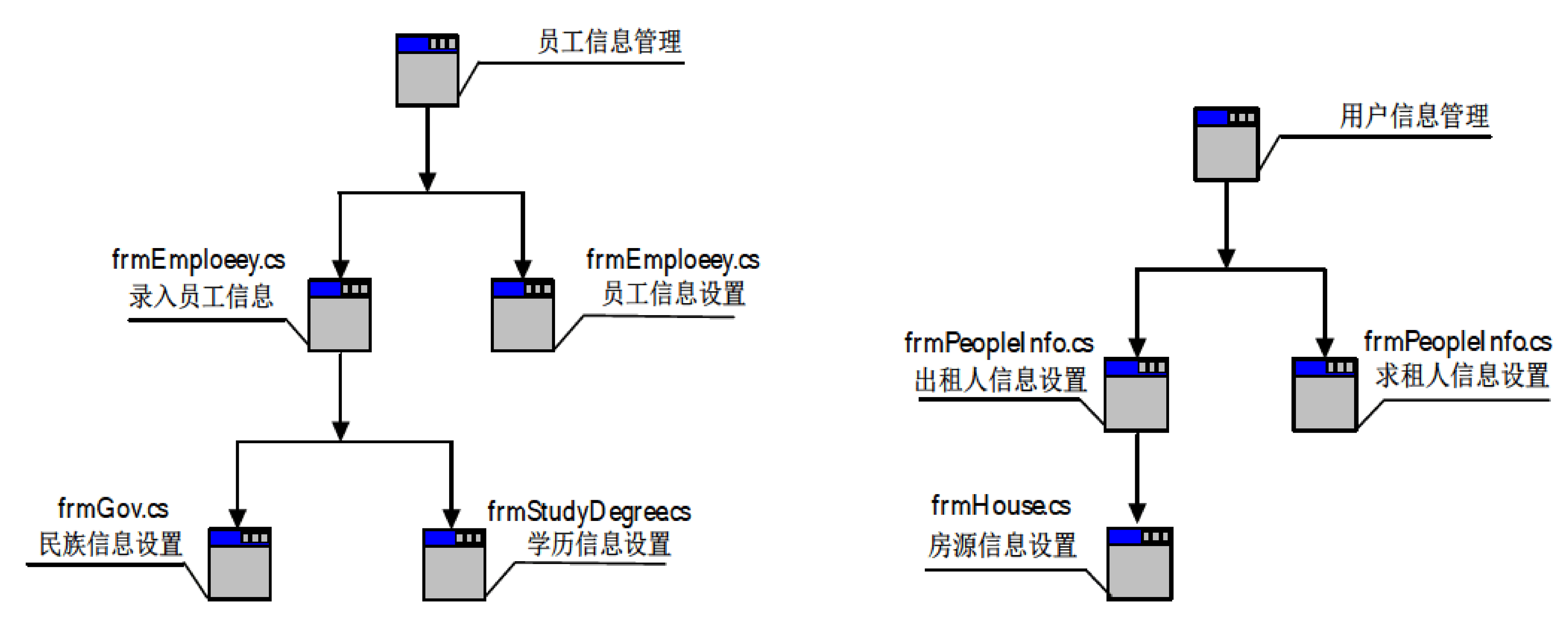


图 23.15　员工信息管理文件架构　　图 23.16　用户信息管理文件架构

求租管理和常用工具文件架构如图 23.17 和图 23.18 所示。

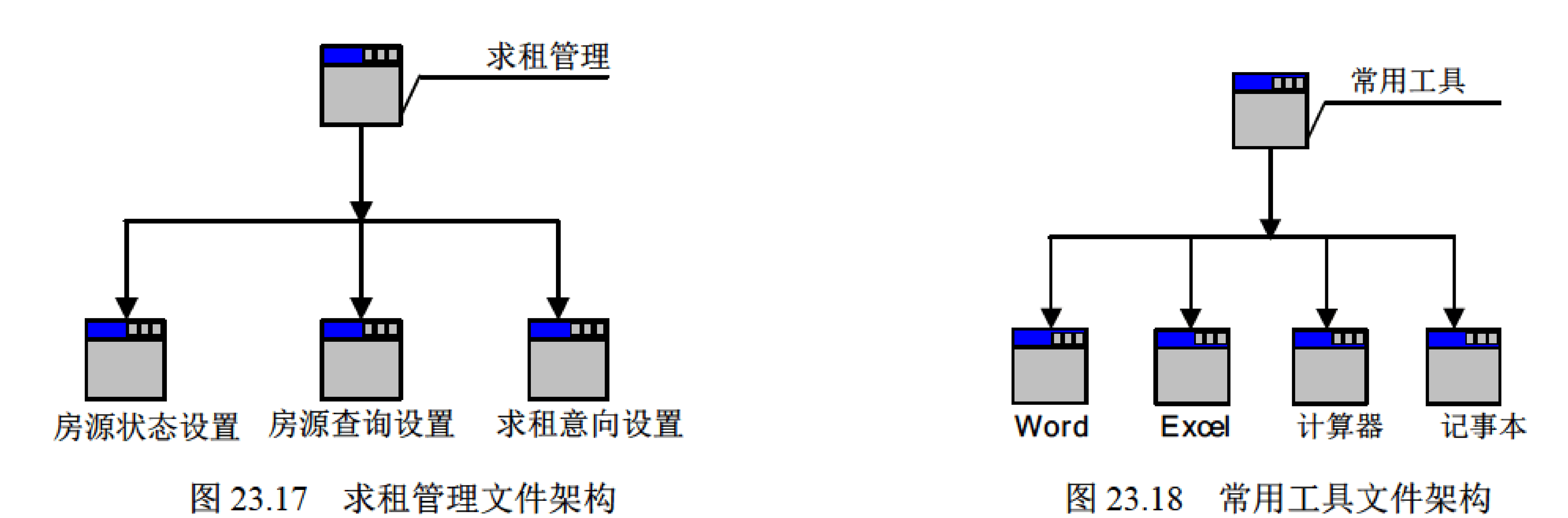


图 23.17　求租管理文件架构　　图 23.18　常用工具文件架构

出租管理文件架构如图 23.19 所示。系统管理文件架构如图 23.20 所示。

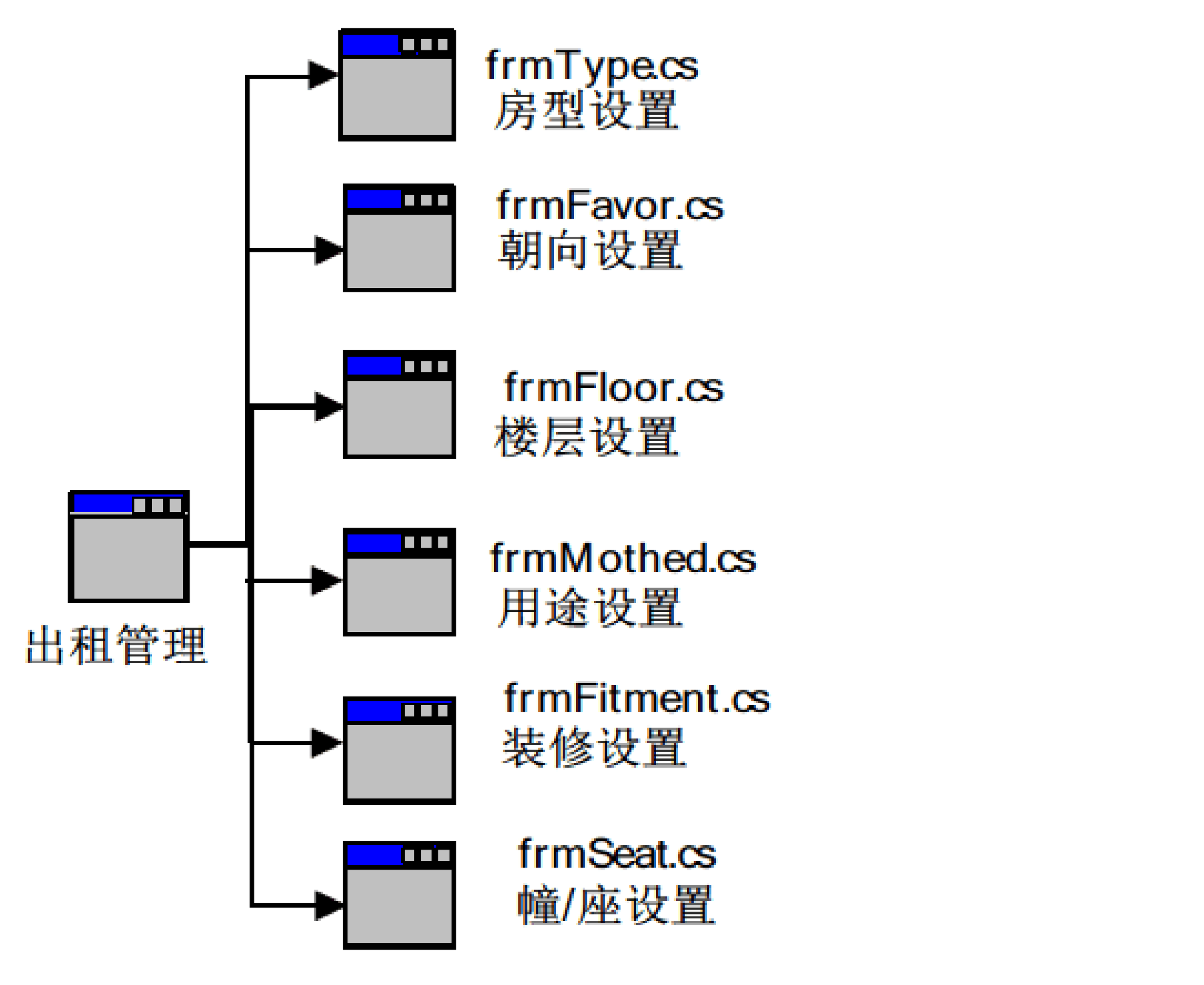



图 23.19　出租管理文件架构

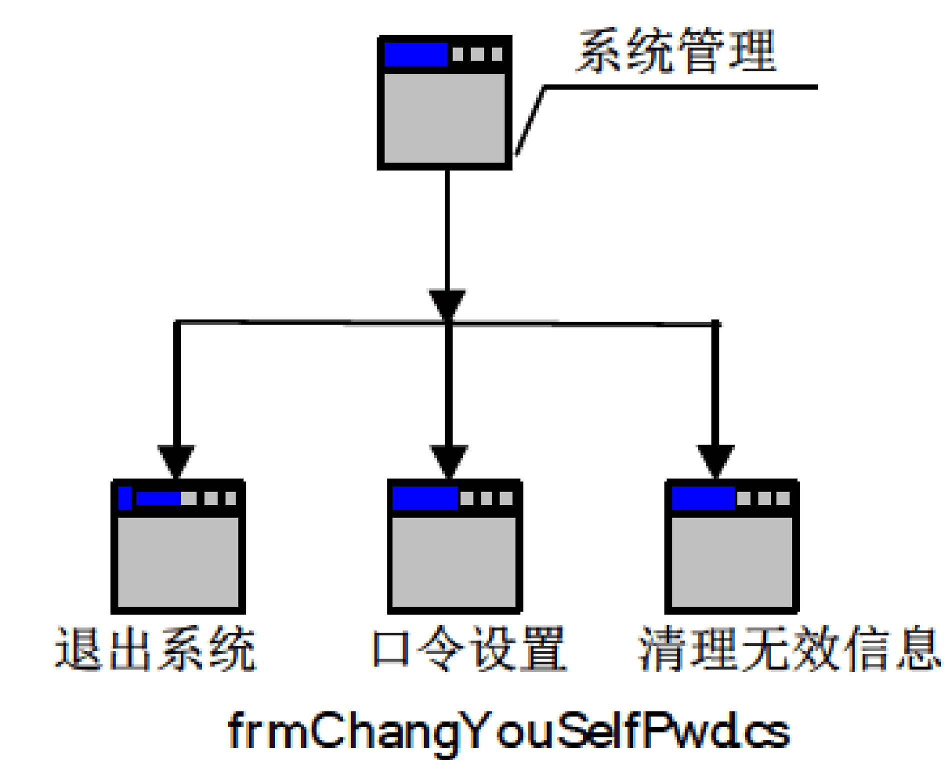


图 23.20　系统管理文件架构

23.5.2　ClsCon 类

ClsCon 主要用于创建数据库连接及关闭打开的数据连接，需要引入 System.Data 和 System.Data.SqlClient 两个命名空间，其关键代码如下：

```
//引用两个命名空间
using System.Data;
using System.Data.SqlClient;
namespace houseAgency.mothedCls
{
    class ClsCon
    {
        …//编写自定义方法
    }
}
```

接下来，对上面代码中的自定义方法进行详细介绍。

1．ConDatebase 方法

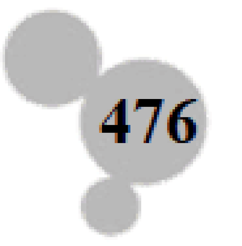
ConDatebase 方法用于建立数据库连接，其实现代码如下：

```
public SqlConnection conn;                                                          //声明 SQL 数据连接引用
public void ConDatabase()                                                           //连接数据库
{
    conn = new SqlConnection("server=.;pwd=;uid=sa;database=db_House");             //创建数据连接对象
}
```

2．closeCon 方法

closeCon 方法实现关闭打开的数据库连接，其实现代码如下：

```
public bool closeCon()                                                              //关闭打开的数据库连接
```



```
{
    try
    {
        if (conn.State == ConnectionState.Open)                    //若数据连接处于打开状态
        {
            conn.Close();                                          //关闭连接

        }
        return true;                                               //返回值为 true
    }
    catch
    {
        return false;                                              //若产生异常，返回值为 false
    }
}
```

23.5.3　clsFavor 类

clsFavor 实体类将 tb_favor 数据表的字段通过 GET、SET 访问器封装起来，其实现代码如下：

```
class clsFavor                                     //定义描述房屋朝向的类
{
    private string house_favorID=null;             //声明表示编号的字符串变量
    private string favor_name=null;                //声明表示名称的字符串变量
    private string favor_remark = null;            //声明表示备注字符串变量
    public string id                               //定义描述房屋朝向编号的属性
    {
        get { return house_favorID; }
        set { house_favorID = value; }
    }
    public string name                             //定义描述房屋朝向名称的属性
    {
        get { return favor_name; }
        set { favor_name = value; }
    }
    public string remark                           //定义描述备注的属性
    {
        get { return favor_remark; }
        set { favor_remark = value; }
    }
}
```

23.5.4　claFavorMethod 类

claFavorMethod 类封装了对 tb_favor 数据表进行插入、修改和删除等操作的方法，由于封装的这 3

种方法在实现技术上类似，所以这里只介绍对 tb_favor 表进行插入操作的方法——insert_table 方法。

insert_table 方法首先通过实体类取出信息，然后调用数据库中的存储过程来得到执行结果，最后把执行结果传递给表示层，其实现代码如下：

```
public string insert_table(clsFavor cf)                                   //实现向朝向数据表插入数据
{
    try
    {
        con.ConDatabase();                                                //创建数据库连接
        SqlCommand cmd = new SqlCommand("proc_favor_insert", con.conn);//创建命令对象
        cmd.CommandType = CommandType.StoredProcedure;                    //表示命令对象将执行存储过程
        cmd.Connection.Open();                                            //打开数据连接
        SqlParameter[] prams =
        {
            new SqlParameter("@house_favorID", SqlDbType.VarChar, 50), //创建朝向编号参数实例
            new SqlParameter("@favor_name", SqlDbType.VarChar, 50),     //创建朝向名称参数实例
            new SqlParameter("@favor_remark", SqlDbType.VarChar, 50),   //创建备注参数实例
            new SqlParameter("@proc_info", SqlDbType.VarChar, 50,        //创建描述执行结果的参数实例
            ParameterDirection.Output,true, 0, 0, string.Empty,DataRowVersion.Default, null)
        };
        prams[0].Value = cf.id;                                           //设置朝向编号
        prams[1].Value = cf.name;                                         //设置朝向名称
        prams[2].Value = cf.remark;                                       //设置备注
        foreach (SqlParameter parameter in prams)                         //添加参数
        {
            cmd.Parameters.Add(parameter);                                //向命令对象中添加参数实例
        }
        cmd.ExecuteNonQuery();                                            //执行存储过程
        string strResult=cmd.Parameters["@proc_info"].Value.ToString();   //获取存储过程的执行结果
        con.closeCon();                                                   //关闭连接
        return strResult;                                                 //返回结果
    }
    catch (Exception ey)
    {
        con.closeCon();                                                   //关闭连接
        return ey.Message.ToString();                                     //返回异常信息
    }
}
```

23.6　主窗体设计

23.6.1　主窗体概述

主窗体是程序操作过程中必不可少的，它是人机交互中的重要环节，用户通过主窗体可以快速打开系统中相关的各个子模块。本系统的主窗体被分为 4 个部分：最上面是系统菜单栏，可以通过它调



用系统中的所有子窗体；菜单栏下面是工具栏，以按钮的形式调用最常用的子窗体；窗体的左面是一个树型菜单，可以通过它显示系统的所有功能；窗体的右面是一张和程序主题相关的背景图片；窗体的最下面，用状态栏显示当前登录的用户名及系统时间。主窗体运行结果如图 23.21 所示。



图 23.21　主窗体

23.6.2　主窗体技术分析

本系统在窗体的左侧使用树型控件（TreeView）显示系统的所有菜单项，通过单击这些菜单项，同样可以打开相应的窗体。这种树型菜单相比传统的横向菜单更加方便和易于操作，下面将介绍在本模块中用到的 TreeView 控件的相关知识。

1．创建 TreeView 控件的根节点

在树型菜单中，根节点显示系统主菜单的内容，那么如何将主菜单的内容添加到 TreeView 控件中呢？这可以通过调用 TreeView 控件的 Nodes 属性的 Add 方法来实现，该方法的重载形式有多种，本模块用到的 Add 方法实现将具有指定标签文本的新树节点添加到当前树节点集合的末尾。语法格式如下：

```
public virtual TreeNode Add(string text);
```

☑　text：节点显示的标签文本。

☑　返回值：添加的节点实例。

例如，下面的示例代码实现向 TreeView 控件添加 3 个表示年级的节点。

```
TreeNode node1 = treeView1.Nodes.Add("一年级");
TreeNode node2 = treeView1.Nodes.Add("二年级");
TreeNode node3 = treeView1.Nodes.Add("三年级");
```

2. 创建根节点的子节点

在树型菜单中，需要在每个根节点下添加子菜单项，这可以通过调用根节点实例的 Nodes 属性的 Add 方法来实现，该方法与上面介绍的 Add 方法是同一个方法，这里不再赘述。

23.6.3 主窗体实现过程

主窗体的具体实现步骤如下。

（1）新建一个 Windows 窗体，命名为 frmMain.cs，它主要用作房屋中介管理系统的主窗体，该窗体主要用到的控件及属性设置如表 23.8 所示。

表 23.8 主窗体主要用到的控件

控件类型	控件 ID	主要属性设置	用途
MenuStrip	menuStrip1	在 Items 属性中设置下拉列表项，并将调用子窗体的菜单项的 Tag 属性，从 1 开始依次设置值	主窗体的下拉列表
ToolStrip	toolStrip1	在 Items 属性中设置按钮项	常用按钮
TreeView	treeView1	BorderStyle 属性设为 FixedSingle, Dock 属性设为 Fill	显示所有子窗体
StatusStrip	statusStrip1	在 Items 属性中设置显示项	显示登录用户名及时间

（2）声明局部变量及公共类 ClsCon 对象，通过 ClsCon 对象调用类中的方法，以实现数据库连接，代码如下：

```
public partial class frmMain : Form
{
    public string M_str_Power = string.Empty;                //定义公共变量，记录登录信息
    string Power = string.Empty;                             //定义私有变量，记录用户权限
    public frmMain ()                                        //窗体的构造器
    {
        InitializeComponent();
    }
    ...//其他事件或方法的代码，可参见本书附带资源包
}
```

在 frmMain 窗体的 Load 事件中，获取登录用户的名称及权限，并通过权限设置“员工信息”菜单的显示状态，然后调用自定义方法 GetMenu 将菜单中的各命令项按照层级关系动态添加到 TreeView 控件中。frmMain 窗体的 Load 事件代码如下：

```
private void frmMain_Load(object sender, EventArgs e)
{
    //在加载时读出权限和用户名信息
    if (M_str_Power != string.Empty)                              //当 M_str_Power 不为空时，即登录成功
    {
        tspname.Text = M_str_Power.Substring(0, M_str_Power.IndexOf('@'));    //获取当前登录的用户名
        tspLoginTime.Text = DateTime.Now.ToLongTimeString();                  //获取当前系统时间
        Power = M_str_Power.Substring(M_str_Power.IndexOf('@') + 1);          //获取当前的用户权限
```



```
        if (Power == "0")                                           //当用户权限为“0”时
        {
            this.tbEmpleey.Visible = false;                         //隐藏“员工信息”菜单
        }
        else
        {
            this.tbEmpleey.Visible = true;                          //显示“员工信息”菜单
        }
        GetMenu(treeView1, menuStrip1);                             //调用自定义方法 GetMenu
    }
}
```

GetMenu 方法是一个无返回值的自定义方法，它的主要功能是遍历 MenuStrip 控件的菜单项，然后将各菜单项按照层级关系添加到 TreeView 控件中。该方法关键代码如下：

```
public void GetMenu(TreeView treeV, MenuStrip MenuS)
{
    for (int i = 0; i < MenuS.Items.Count; i++)                     //遍历 MenuStrip 组件中的一级菜单项
    {
        //将一级菜单项的名称添加到 TreeView 组件的根节点中，并设置当前节点的子节点 newNode1
        TreeNode newNode1 = treeV.Nodes.Add(MenuS.Items[i].Text);
        newNode1.Tag = 0;
        //将当前菜单项的所有相关信息存入到 ToolStripDropDownItem 对象中
        ToolStripDropDownItem newmenu = (ToolStripDropDownItem)MenuS.Items[i];
        //判断当前菜单项中是否有二级菜单项
        if (newmenu.HasDropDownItems && newmenu.DropDownItems.Count > 0)
            for (int j = 0; j < newmenu.DropDownItems.Count; j++)      //遍历二级菜单项
            {
                //将二级菜单名称添加到 TreeView 的子节点 newNode1 中，并设置当前节点的子节点 newNode2
                TreeNode newNode2 = newNode1.Nodes.Add(newmenu.DropDownItems[j].Text);
                //将菜单项的 Tag 属性值赋给当前节点的 Tag 属性，便于打开相应的子窗体
                newNode2.Tag = int.Parse(newmenu.DropDownItems[j].Tag.ToString());
                //将当前菜单项的所有相关信息存入到 ToolStripDropDownItem 对象中
                ToolStripDropDownItem newmenu2 = (ToolStripDropDownItem)newmenu.DropDownItems[j];
            }
    }
}
```

因为本系统既可以在菜单栏中打开子窗体，又可以在树型菜单中打开窗体，所以要设置一个自定义方法 frm_show，通过各菜单项或节点的 Tag 属性值来打开相应的窗体。

```
public void frm_show(int n)
{
    switch (n)                                          //通过标识调用各子窗体
    {
        case 0: break;
        case 1:
```

```
                {
                    frmPeopleInfo fp = new frmPeopleInfo();       //实例化一个求租人信息窗体
                    fp.strID = "want";                            //设置窗体中的公共变量，表示求租
                    fp.Text = "求租人员信息";                      //设置窗体名称
                    fp.ShowDialog();                              //用对话框模式打开窗体
                    fp.Dispose();                                 //释放窗体的所有资源
                    break;
                }
            case 2:
                {
                    frmPeopleInfo fp = new frmPeopleInfo();       //实例化一个出租人信息窗体
                    fp.strID = "lend";                            //设置窗体中的公共变量，表示出租
                    fp.Text = "出租人员信息设置";                  //设置窗体名称
                    fp.ShowDialog();                              //打开模式对话框窗体
                    fp.Dispose();                                 //释放资源
                    break;
                }
            case 3:
                {
                    frmPeopleList fp = new frmPeopleList();       //实例化一个用户信息管理窗体
                    fp.ShowDialog();                              //以对话框模式打开窗体
                    fp.Dispose();                                 //释放资源
                    break;
                }
             …//因为篇幅有限只给出部分代码
            case 26:
                {
                    if (MessageBox.Show("确认退出系统吗？ ", "提示", MessageBoxButtons
.OKCancel, MessageBoxIcon.Question) == DialogResult.OK)   //若确认退出
                        Application.Exit();                       //关闭当前应用程序
                    break;
                }
            case 27:
                {
                    frmStock fs = new frmStock();                 //实例化一个备份数据窗体
                    fs.ShowDialog();                              //以对话框模式打开窗体
                    fs.Dispose();                                 //释放资源
                    break;
                }
            case 28:
                {
                    frmRestore fr = new frmRestore();             //实例化一个还原数据窗体
                    fr.ShowDialog();                              //以对话框模式打开窗体
                    fr.Dispose();                                 //释放资源
                    break;
                }
            case 29:
                {
                    ClsCon con = new ClsCon();                        //实例化一个 ClsCon 公共类
```

```
                    con.ConDatabase();                              //连接数据库
                    //清理出租人和房源之间的垃圾信息
                    //如当出租人要出租房时，可是没有给出房源信息，这时出租人信息就没有用了
                    try
                    {
                        SqlCommand cmd = new SqlCommand();          //实例化一个 SqlCommand 对象
                        cmd.Connection = con.conn;                  //与数据库建立连接
                        cmd.Connection.Open();                      //打开数据库的连接
                        cmd.CommandText = "proc_clear";             //存储过程的名
                        cmd.CommandType = CommandType.StoredProcedure;
                        cmd.ExecuteNonQuery();                      //执行存储过程
                        con.closeCon();                             //关闭数据库的连接
                        MessageBox.Show("恭喜已清除！！！");
                    }
                    catch (Exception ey)
                    {
                        MessageBox.Show(ey.Message);                //弹出异常信息提示框
                    }
                    break;
                }
            case 30:
                {
                    MessageBox.Show("\t 你可以到明日科技网站\t\n\n\t   得到你想知
道的\n\t      谢谢使用！！");                                    //打开帮助对话框
                    break;
                }
        }
}
```

下面用“求租人员信息”命令的单击事件为例，来说明如何用自定义方法 frm_show 来调用相应的子窗体。“求租人员信息”命令的 Click 事件关键代码如下：

```
private void  求租人员信息 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frm_show(int.Parse(((ToolStripMenuItem)sender).Tag.ToString()));          //打开“求租人员信息”窗体
}
```

23.7　用户信息管理模块设计

23.7.1　用户信息管理模块概述

用户信息管理主要用于管理用户信息。其中包括两种用户类型，即出租方和求租方。如果出租方仅仅提供个人基本信息而没有提供房源信息，则可以通过本系统提供的垃圾信息清理机制将其清除。用户信息管理窗体运行结果如图 23.22 所示。

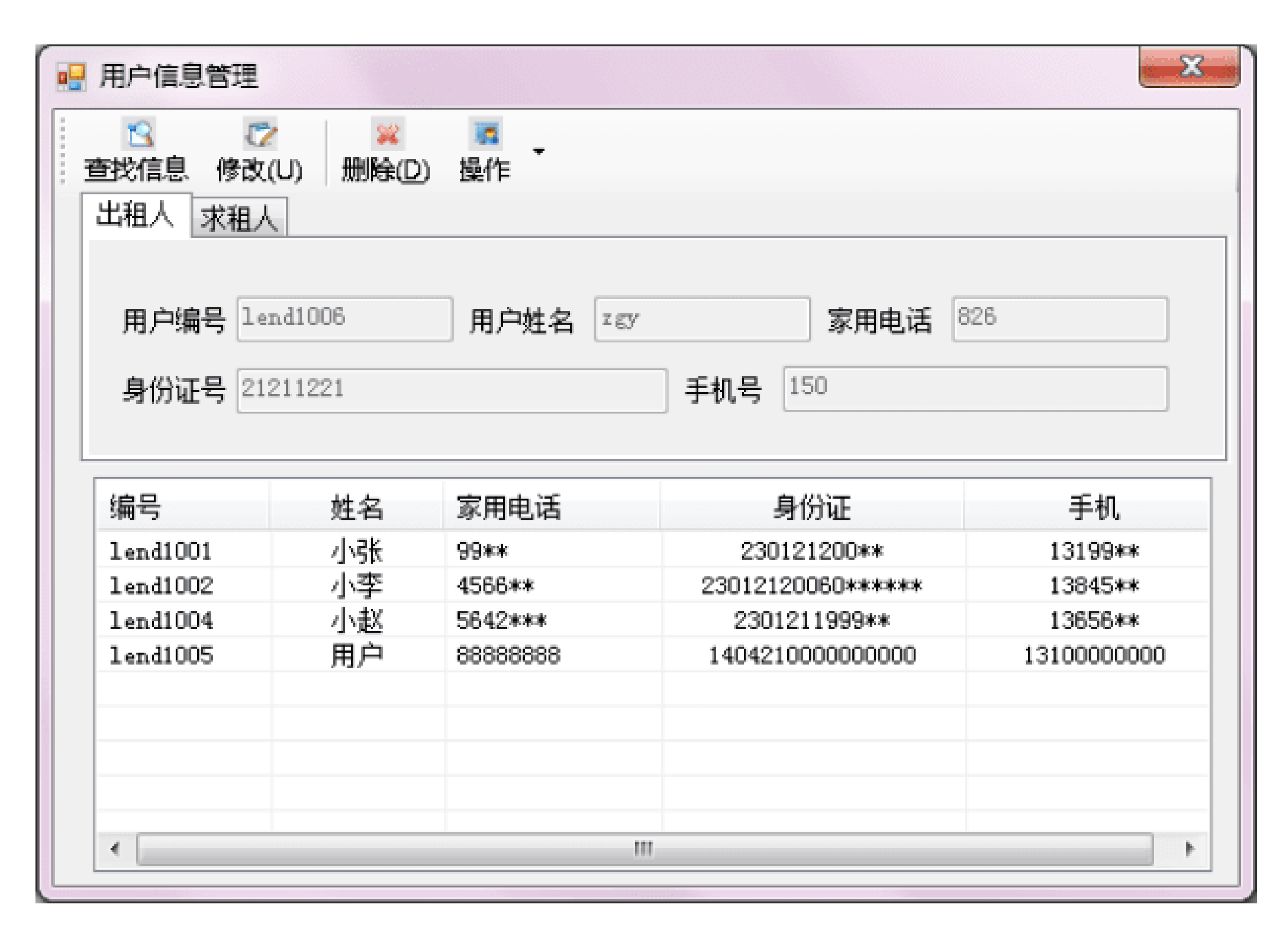


图 23.22　用户信息管理窗体

23.7.2　用户信息管理模块技术分析

本模块提供了查询“出租人”和“求租人”的功能，并且可以通过设置多个查询条件来查询，这就需要动态设置具有查询功能的 SQL 语句，本实例使用 StringBuilder 类的 Append 方法来实现动态连接 SQL 语句，该方法的重载形式有多种，本模块用到的 Append 方法实现在 StringBuilder 类型实例的结尾追加指定字符串的副本。语法格式如下：

```
public StringBuilder Append(string value);
```

☑　value：要追加的字符串。

☑　返回值：完成追加操作后对 StringBuilder 类型实例的引用。

例如，下面的代码实现根据客户编号、客户名称、客户电话号码等多个条件实现动态查询“求租人”或“出租人”记录。

```
StringBuilder sbSql = new StringBuilder(" select * from tb_User ");
sbSql.Append(" where User_IDs like '%" + this.textBox1.Text.ToString() + "%'");
sbSql.Append(" and User_names like '%" + this.textBox2.Text.ToString() + "%'");
sbSql.Append(" and User_phone like '%" + this.textBox5.Text.ToString() + "%'");
sbSql.Append(" and User_cardID like '%" + this.textBox4.Text.ToString() + "%'");
sbSql.Append(" and User_homePhone like '%" + this.textBox3.Text.ToString() + "%'");
```

23.7.3　用户信息管理模块实现过程

用户信息管理模块的具体实现步骤如下。

（1）新建一个 Windows 窗体，命名为 frmPeopleList.cs，用于设置用户信息。该窗体主要用到的控件及属性设置如表 23.9 所示。

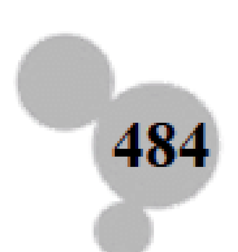


表 23.9　用户信息管理窗体主要用到的控件

控件类型	控件 ID	主要属性设置	用　途
TextBox	txtID	将其 ReadOnly 属性设置为 false	用户编号
	txtName	同上	用户姓名
	txtHomePhome	同上	家用电话
	txtPhone	同上	手机号
	txtCardID	同上	身份证号
ToolStrip	toolStrip1	Items 属性获取属于 ToolStrip 的所有项； TextDirection 属性获取或设置在 ToolStrip 属性上绘制文本的方向； ImageList 属性获取或设置 ToolStrip 项上显示的图像的图像列表； ImageScalingSize 属性获取或设置 ToolStrip 上所用图像的大小，以像素为单位	控制操作
ListView	listView1	Columns 属性用于设置“详细信息”视图中显示的列	显示用户信息
TabControl	tabControl1	TabPages 属性表示 TabControl 控件的所有选项卡；Alignment 属性用于设置选项卡的显示部位	作为容器

（2）声明局部变量及公共类 ClsCon 的对象，通过该对象调用类中的方法，以实现数据库连接，代码如下：

```
namespace houseAgency
{
    public partial class frmPeopleList : Form
    {
        StringBuilder sbSql = new StringBuilder();                    //用于存放 SQL 语句头
         StringBuilder sbWhere = new StringBuilder();                 //用于生成 SQL 语句的条件
         StringBuilder sbWhereInfo = new StringBuilder();             //用于生成 SQL 语句的条件
         ClsCon con = new ClsCon();                                   //连接对象
         string strTemp = string.Empty;                               //临时变量
         public frmPeopleList()                                       //构造方法
         {
             InitializeComponent();
             con.ConDatabase();                                       //连接数据库
         }
         …//其他事件或方法代码，参见本书附带资源包
    }
}
```

在 frmPeopleList 窗体的 Load 事件中，通过调用自定义 ListInfo 方法对 ListView 控件进行数据绑定，显示所有系统用户信息。窗体 Load 事件关键代码如下：

```
private void frmPeopleList_Load(object sender, EventArgs e)
{
    con.ConDatabase();                                           //创建数据库连接
    sbSql.Append("select User_IDs,User_names,User_homePhone,User_cardID,User_phone from tb_User");
    ListInfo(sbSql.ToString());                                  //将显示信息绑定到 ListView 控件
```

```
    UnAble();
}
```

自定义 UnAble 方法，主要用来批量设置容器控件中相关控件的 Enabled 属性。代码如下：

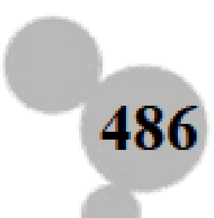
```
private void UnAble()
{
    foreach (Control ct in this.tabPage1.Controls)                //遍历 tabPage1 中的所有控件
    {
        //如果当前控件是 TextBox
        if (ct.GetType().ToString() == "System.Windows.Forms.TextBox")
            ct.Enabled = false;                                   //设置该控件为不可用状态
    }
    foreach (Control ctT in this.tabPage2.Controls)               //遍历 tabPage2 中的所有控件
    {
        //如果当前控件是 TextBox
        if (ctT.GetType().ToString() == "System.Windows.Forms.TextBox")
            ctT.Enabled = false;                                  //设置该控件为不可用状态
    }
}
```

自定义 ListInfo 方法，该方法接受查询语句，用来将查询结果绑定到 ListView 控件。代码如下：

```
private void ListInfo(string SQL)
{
    con.ConDatabase();                                            //连接数据库
    this.listView1.Items.Clear();                                 //清空 listView1 控件
    SqlDataAdapter da = new SqlDataAdapter(SQL, con.conn);//实例化 SqlDataAdapter 类
    DataTable dt = new DataTable();                               //实例化 DataTable 对象
    //通过 SqlDataAdapter 对象的 Fill 方法，将数据表信息添加到 DataTable 对象中
    da.Fill(dt);
    foreach (DataRow dr in dt.Rows)                               //遍历所有行
    {
        ListViewItem lv;                                          //实例一个项
        lv = new ListViewItem(dr[0].ToString());                  //添加第 1 个字段值
        lv.SubItems.Add(dr[1].ToString());                        //添加第 2 个字段值
        lv.SubItems.Add(dr[2].ToString());                        //添加第 3 个字段值
        lv.SubItems.Add(dr[3].ToString());                        //添加第 4 个字段值
        lv.SubItems.Add(dr[4].ToString());                        //添加第 5 个字段值
        this.listView1.Items.Add(lv);                             //在控件中添加当前项，也就是行记录
    }
}
```

单击 ListView 控件中的任一单元格，将对应的详细客户信息显示在相应选项卡的文本框中。实现代码如下：

```
private void listView1_Click(object sender, EventArgs e)
{
```



```
    con.ConDatabase();                                                //连接数据库
    string strID =this.listView1.SelectedItems[0].Text.ToString();   //获取选中项信息的 ID 号
    string sql = "select User_IDs,User_names,User_homePhone,User_cardID,
    User_phone from tb_User where user_ids='" + strID + "'";          //查找的 SQL 语句
    SqlCommand cmd=new SqlCommand(sql,con.conn);                      //执行 SQL 语句
    con.closeCon();                                                   //关闭当前连接
    cmd.Connection.Open();                                            //打开数据库连接
    SqlDataReader dr = cmd.ExecuteReader();                           //读取表中的信息
    if (strID.Substring(0, 4) == "lend")                              //当编号的前 4 个字符是"lend"时，表示出租人
    {
        while (dr.Read())                                             //循环读取行信息
        {
            //将行信息的内容添加到 textBox1 上
            this.textBox1.Text = dr[0].ToString();
            this.textBox2.Text = dr[1].ToString();
            this.textBox3.Text = dr[2].ToString();
            this.textBox4.Text = dr[3].ToString();
            this.textBox5.Text = dr[4].ToString();
        }
        this.tabControl1.SelectTab(0);                                //令出租人选项卡为当前页
    }
    else
    {
        while (dr.Read())                                             //循环读取数据
        {
            //将行信息的内容添加到 textBox1 上
            this.textBox10.Text = dr[0].ToString();
            this.textBox9.Text = dr[1].ToString();
            this.textBox8.Text = dr[2].ToString();
            this.textBox7.Text = dr[3].ToString();
            this.textBox6.Text = dr[4].ToString();
        }
        this.tabControl1.SelectTab(1);                                //令求租人选项卡为当前页
    }
    dr.Close();                                                       //关闭数据表
    con.closeCon();                                                   //关闭连接
    tb_update.Enabled = true;                                         //使该控件可用
}
```

当用户单击“出租人”选项卡或“求租人”选项卡时，在相应的选项卡页中显示客户信息。实现代码如下：

```
private void tabControl1_SelectedIndexChanged(object sender, EventArgs e)
{
    if (this.tabControl1.SelectedTab.Text == "出租人")          //如果当前选中的选项卡是"出租人"
    {
```

```
        sbWhere.Append(" where user_type='lend'");          //将查询条件添加到 sbWhere 实例中
        ListInfo(sbSql.ToString() + sbWhere.ToString());    //调用自定义方法 ListInfo
        sbWhere.Remove(0, sbWhere.Length);                  //移除当前字例中的内容
    }
    else if (this.tabControl1.SelectedTab.Text == "求租人")   //如果当前选中的选项卡是"求租人"
    {
        sbWhere.Append(" where user_type='want' ");         //将查询条件添加到 sbWhere 实例中
        ListInfo(sbSql.ToString() + sbWhere.ToString());    //调用自定义方法 ListInfo
        sbWhere.Remove(0, sbWhere.Length);                  //移除当前字例中的内容
    }
}
```

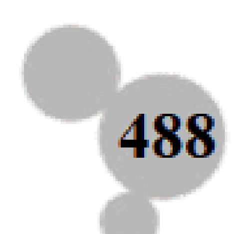
单击“删除”按钮，删除相关的客户信息，同时自动调用触发器 trig_delete_tbUser，删除出租人所提供的房源信息。程序中实现代码如下：

```
private void tb_delete_Click(object sender, EventArgs e)
{
    con.ConDatabase();                                          //连接数据库
    //调用触发器删除用户时去删它对应的房源信息
    if (MessageBox.Show("是否删除用户？ ", "提示", MessageBoxButtons.YesNo,
    MessageBoxIcon.Hand) == DialogResult.Yes)                   //若确认删除
    {
        //实例化 SqlCommand 对象
          SqlCommand cmd = new SqlCommand("delete from tb_user where User_IDS='"
        +this.listView1.SelectedItems[0].Text.ToString()+ "'", con.conn);
        cmd.Connection.Open();                                  //打开连接
        cmd.ExecuteNonQuery();                                  //执行 SQL 的删除语句
        con.conn.Close();                                       //关闭数据库连接
        ListInfo(sbSql.ToString());                             //利用自定义方法 ListInfo 更新数据
    }
}
```

23.8　房源设置模块设计

23.8.1　房源设置模块概述

房源设置用于设置房源的基本信息，它将多个基础表的信息和房源表的信息进行有机的结合。通过视图 view_house 把信息呈现给用户。本系统较为人性化的功能也在这里体现，即出租人在添加房源信息完毕时，程序通过存储过程 proc_house_insert 为出租人查找匹配的意向求租信息。如果有符合的信息，则会显示出来，出租人可以根据显示的求租信息找到合适的求租人，这样大大提高了工作效率。房源设置窗体运行结果如图 23.23 所示。



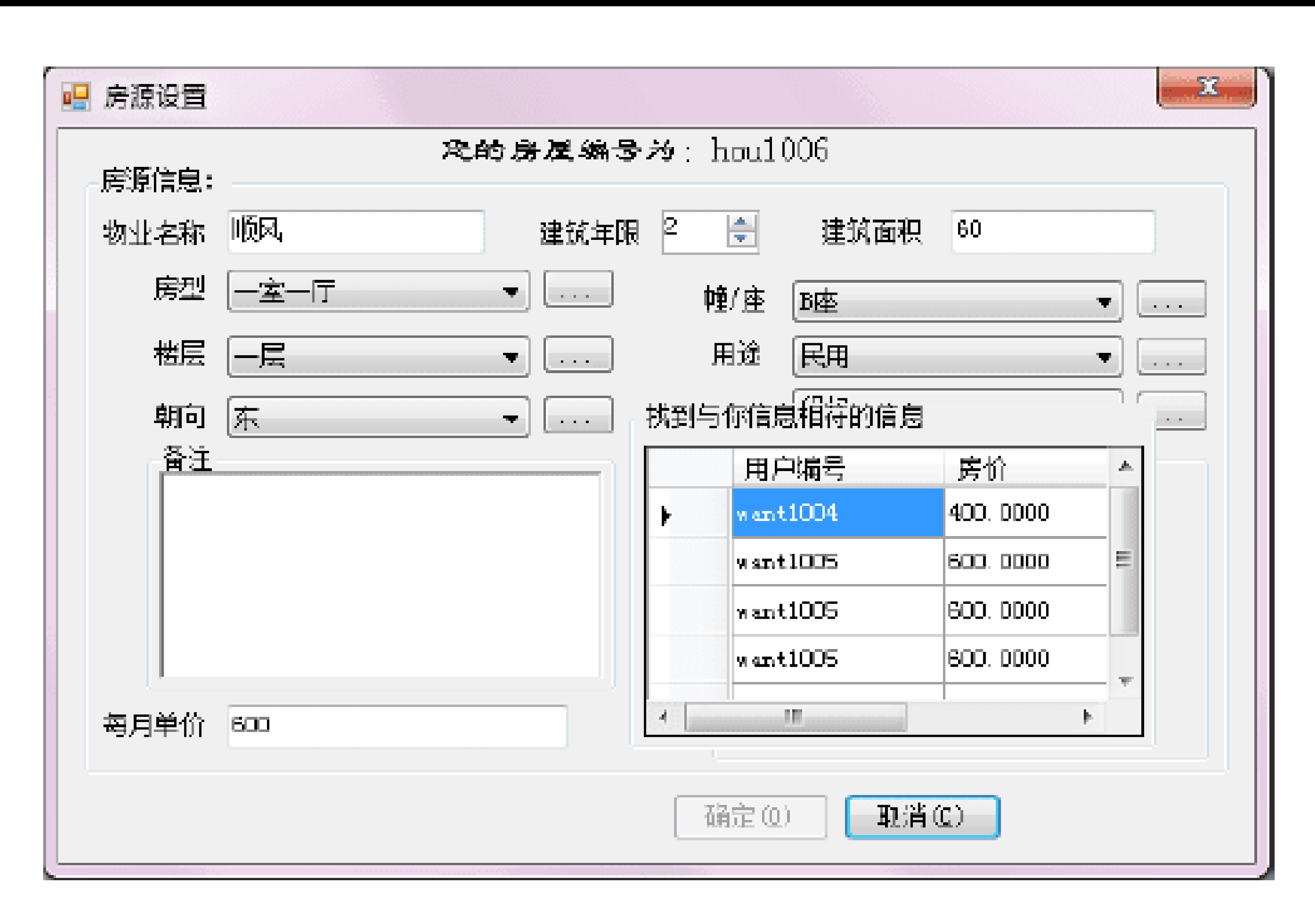


图 23.23　房源设置窗体

23.8.2　房源设置模块技术分析

在房源设置窗体的最上方会显示房屋编号，该房屋编号的规则是：以字符串 hou 为编号前缀，加上 4 位数字作为编号的后缀，这 4 位数字从 1001 开始。例如，在添加第一个房源信息时，房屋编号就是 hou1001，接下来的其他房屋编号会在上一个最大房屋编号的后缀的基础上进行流水递增，这样程序就需要取出已存在的最大房屋编号。在 SQL 语句中，使用 Max(house_ID)函数获取最大房屋编号。在 C#程序中，使用 SqlCommand 实例的 ExecuteScalar 方法获取该房屋编号，下面将介绍 ExecuteScalar 方法。

ExecuteScalar 方法执行指定的 SQL 查询，并返回查询所返回的结果集中第一行的第一列。语法格式如下：

```
public override object ExecuteScalar();
```

该方法返回结果集中第一行的第一列；如果结果集为空，则为空引用。

例如，下面的示例代码通过 ExecuteScalar 方法获取最大的房源编号。

```
SqlCommand cmd = new SqlCommand("select Max(house_ID) from tb_house", con.conn);  //创建命令对象
cmd.Connection.Open();                                                              //打开数据库连接
strResult = cmd.ExecuteScalar().ToString();                                         //获取最大房源编号
```

23.8.3　房源设置模块实现过程

房源设置模块的具体实现步骤如下。

（1）新建一个 Windows 窗体，命名为 frmHouse.cs，用于设置房屋信息，该窗体主要用到的控件及属性设置如表 23.10 所示。

表 23.10　房源设置窗体主要用到的控件

控件类型	控件 ID	主要属性设置	用　途
abl TextBox	txtName	将其 ReadOnly 属性设置为 false	物业名称
	txtArea	同上	建筑面积
	txtPrice	同上	每月单价

续表

控 件 类 型	控件 ID	主要属性设置	用　　途
ComboBox	cobFlood	将其 DropDownStyle 属性设置为 DropDownList	楼层
	cobFavoe	同上	朝向
	cobXing	同上	房型
	cobZhuang	同上	装修
	cobDong	同上	幢/座
	cobUser	同上	用途
Button	btnSelect	TextAlign 属性值共有 9 种，这里设置为居中 MiddleCenter	确定（添加）
	btnClear	同上	取消
	btnUpdate	同上	修改
	btnOK	同上	就租你了（选定房源）
DataGridView	dgvResult	设置 SelectionMode 属性为 FullRowSelect，即选取整行	显示求租意向信息
OpenFileDialog	opImage	Filter 用于筛选文件类型	选取图片

（2）声明局部变量及公共类 ClsCon 的对象，通过该对象调用类中的方法，以实现数据库连接，实现代码如下：

```
public partial class frmHouse : Form
{
    public string M_str_Show = String.Empty;              //定义表示录入或浏览数据的标记
    public string M_str_temp = string.Empty;              //定义表示空的临时变量
    string strResult = string.Empty;                      //定义存储最大房源编号的变量
    string strPath = string.Empty;                        //定义存储房源图片路径的变量
    string strSatae = string.Empty;                       //定义表示数据提交状态的标记
    ClsCon con=new ClsCon();                              //实例化公共类 ClsCon
    ClsHouse ch = new ClsHouse();                         //实例化公共类 ClsHouse
    ClsHouseMethed chm = new ClsHouseMethed();            //实例化公共类 ClsHouseMethed
    public frmHouse()
    {
        InitializeComponent();
    }
    …//其他事件或方法的代码
}
```

在 frmHouse 窗体的 Load 事件中，通过 M_str_Show 变量判断本次调用窗体的目的。如果是浏览或修改信息，则将相应的信息显示到控件上；如果是添加信息，则将基本表的信息绑定到 ComboBox 控件上。frmHouse 窗体的 Load 事件中实现代码如下：

```
private void frmHouse_Load(object sender, EventArgs e)
{
    string strHouseState = string.Empty;                  //定义字符串变量
    con.ConDatabase();                                    //连接数据库
```



```
 //根据自定义方法获取指定表的数据
flushFaove(); flushfitment(); flushfloor();
flushmothed(); flushseat(); flushtype();
if (M_str_Show == String.Empty)                          //若为空，则表示插入房源操作
{
    try
    {//实例化 SqlCommand 对象
          SqlCommand cmd = new SqlCommand("select Max(house_ID) from tb_house", con.conn);
        cmd.Connection.Open();                          //打数据库连接
        strResult = cmd.ExecuteScalar().ToString();     //执行 SQL 语句
        con.closeCon();                                 //关闭数据库的连接
        if (strResult == "")                            //如果查询为空
        {
            strResult = "hou1001";                      //设置第一个 ID 号
        }
        else
        {
            string strTemp = strResult.Substring(3); //获取 ID 中的编码
              //设置要添加信息的 ID 号
            strResult = "hou" + Convert.ToString(Int32.Parse(strTemp) + 1);
        }
        this.lblHouseID.Text = "您的房屋编号为：" + strResult; //显示添加信息的 ID 号
    }
    catch (Exception ey)
     {
         con.closeCon();                                //关闭数据库连接
         MessageBox.Show(ey.Message);
     }
 }
 else
 {
     this.button8.Visible = false;                      //隐藏“取消”按钮
     this.butOK.Visible = false;                        //隐藏“确定”按钮
     Visable();//设置
     SqlCommand cmd = new SqlCommand("select * from tb_house where house_ID='" +
     M_str_Show + "' ", con.conn);                      //创建命令对象
     con.conn.Open();                                   //打开数据连接
     SqlDataReader dr = cmd.ExecuteReader();            //执行 SQL 命令
     if (dr.HasRows)                                    //判断是否有记录
     {
         while (dr.Read())                              //遍历表中的行信息
         {
             //将当前行中的各字段信息添加到指定控件中
             lblHouseID.Text = dr[0].ToString();
             this.txtName.Text = dr[1].ToString();
             this.picHouse.ImageLocation = dr[8].ToString();
             txtPrice.Text = dr[9].ToString();
             this.nudYear.Value = Convert.ToDecimal(dr[11].ToString());
             this.txtArea.Text = dr[12].ToString();
             this.ttbRemark.Text = dr[13].ToString();
             strHouseState = dr[4].ToString();
             this.cboXing.SelectedValue = dr[2].ToString();
```

```
                    this.cobDong.SelectedValue = dr[3].ToString();
                    this.cboFavoe.SelectedValue = dr[6].ToString();
                    this.cobZhuang.SelectedValue = dr[5].ToString();
                    this.cobUser.SelectedValue = dr[7].ToString();
                    this.cobFlood.SelectedValue = dr[10].ToString();
                }
            }
            con.closeCon();                                      //关闭数据库的连接
            if (strHouseState == "none")
            {
                button1.Visible = true;                          //显示该控件
                button2.Visible = true;
            }
}
```

输入房源信息时，为了保证建筑面积和单价信息的有效性，在 TextBox 的 KeyPress 事件中调用自定义 IsNum 方法，该方法用来验证用户输入建筑面积和单价信息的合法性。自定义 IsNum 方法的代码如下：

```
private void IsNum(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 8)                                          //退格键
    {
        return;
    }
    else if (e.KeyChar == 13)                                    //回车键
    {
        SendKeys.Send("{Tab}");
    }
    else if (e.KeyChar > '9' || e.KeyChar < '0' && e.KeyChar != '.')  //数字与小数点
    {
        e.Handled = true;                                        //不处理当前操作
        MessageBox.Show("无效字符");
    }
}
```

在图 23.23 中所示的窗体中单击“…”按钮进行更改相应的基础信息，在确认更改后，新的基础信息会立即加载到相应的 ComboBox 控件中。这里以“更改房型”为例，其实现代码如下：

```
private void flushtype()                                         //实现刷新房型信息
{
    con.ConDatabase();                                           //获取数据库连接
    try
    {
        //实例化 SqlDataAdapter 对象
          SqlDataAdapter da = new SqlDataAdapter("select * from tb_type", con.conn);
        DataTable dt = new DataTable();                          //实例化 DataTable 对象
          //通过 SqlDataAdapter 对象的 Fill 方法，将数据表信息添加到 DataSet 对象中
        da.Fill(dt);                                             //填充 DataTable 实例
        cboXing.DataSource = dt.DefaultView;                     //控件绑定到数据源
```

```
            cboXing.DisplayMember = "type_names";                   //设置显示值
            cboXing.ValueMember = "huose_typeID";                   //设置数据值
        }
        catch (Exception ey)
        {
            MessageBox.Show(ey.Message);                            //输出异常信息
        }
}
private void button1_Click(object sender, EventArgs e)
{
        frmType ft = new frmType();                                 //实例化 frmType 窗体
        if (ft.ShowDialog() == DialogResult.OK)                     //打开当前窗体
        {
            flushtype();                                            //调用方法 flushtype 刷新房型信息
        }
}
```

23.9　房源信息查询模块设计

23.9.1　房源信息查询模块概述

房源信息查询是房屋中介系统中重要的功能之一，它主要根据物业名称、楼层、价格、面积、朝向等条件进行查询，并且部分字段支持模糊查询。房源信息查询窗体运行结果如图 23.24 所示。



图 23.24　房源信息查询窗体

23.9.2　房源信息查询模块技术分析

房源信息查询窗体是将本窗体中的各个查询条件组合为 SQL 查询语句，然后在指定的数据表中进行查询。

下面对 SQL 的查询语句进行详细说明。

```
SELECT select_list [ FROM table_source ][ WHERE search_condition ]
```

☑ select_list：数据表中的字段名称，可以用*表示所有字段。
☑ table_source：数据表名称。
☑ search_condition：条件表达式。

本模块应用 SqlDataAdapter 对象来执行 SQL 查询语句，其语法格式如下：

```
SqlDataAdapter(string selectCommandText, SqlConnection selectConnection);
```

☑ selectCommandText：SQL 语句。
☑ selectConnection：表示 SQL Server 数据库的一个打开连接。

下面用 SqlDataAdapter 对象实现一个简单的数据表查询功能。代码如下：

```
SqlDataAdapter da = new SqlDataAdapter("select * from view_house", con.conn);
DataTable dt = new DataTable();
//通过 SqlDataAdapter 对象的 Fill 方法，将数据表信息添加到 DataSet 对象中
da.Fill(dt);
this.dataGridView1.DataSource = dt.DefaultView;     //用 dataGridView1 控件显示表信息
```

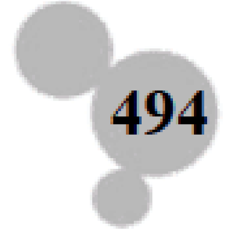
23.9.3 房源信息查询模块实现过程

房源信息查询模块的具体实现步骤如下。

（1）新建一个 Windows 窗体，命名为 frmSelect.cs，用于查询房源信息，该窗体主要用到的控件及属性设置如表 23.11 所示。

表 23.11 房源信息查询窗体主要用到的控件

控件类型	控件名称	主要属性设置	用途
TextBox	txtName	将其 ReadOnly 属性设置为 false	物业名称
	txtArea	同上	建筑面积
	txtPrice	同上	价钱
	txtHuoseID	同上	房屋编号
ComboBox	cobFlood	将其 DropDownStyle 属性设置为 DropDownList	楼层
	cobFavoe	同上	朝向
	cobXing	同上	房型
	cobZhuang	同上	装修
	cobDong	同上	幢/座
	cobUser	同上	用途
Button	btnSelect	将 TextAlign 属性设置为 MiddleCenter；将 UseMnemonic 属性设为 true，这样“_”符号后面的第一个字符将用作标签的助记键	查询
	btnClear	同上	清空
	btnSelectAll	同上	全选



续表

控件类型	控件名称	主要属性设置	用　途
DataGridView	dataGridView1	SelectionMode 属性设置为 FullRowSelect，以选取整行；单击 RowTemplate 属性列表选择 DefaultCellStyle 属性，将出现 CellStyle 生成器，在该生成器内选择 SelectionBackColor 属性，以设置被选取行的前景颜色	显示房源信息
ErrorProvider	epIfo	BlinkStyle 属性设置为 BlinkIfDifferentError 该属性用于控制当确定错误后，错误图标是否闪烁	提示错误信息
NumericUpDown	nudYear	Minimum 和 Maxinmum 属性用于设置最小值和最大值，这里设置为 1 和 100	显示建筑年限
CheckBox	chkCheck	CheckState 属性设置为 Unchecked	控制查询条件

（2）声明局部变量及公共类 ClsCon 的对象，通过 ClsCon 的对象调用类中的方法，用于实现数据库连接，实现代码如下：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
using houseAgency.mothedCls;
namespace houseAgency
{
    public partial class frmSelect : Form
    {
        StringBuilder strSql = new StringBuilder();
        string strMidle = string.Empty;
        string strWhere = string.Empty;
        ClsCon con = new ClsCon();
        public frmSelect()
        {
            InitializeComponent();
        }
        …//其他事件或方法代码，参见本书附带资源包
    }
}
```

在 frmSelect 窗体的 Load 事件中，DataGridView 控件进行数据绑定，以显示房源相关信息。frmSelect 窗体的 Load 事件实现代码如下：

```
private void frmSelect_Load(object sender, EventArgs e)
{
   try
```

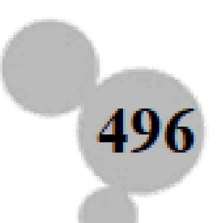
```
    {
        con.ConDatabase();
        SqlDataAdapter da = new SqlDataAdapter("select * from view_house", con.conn);
        DataTable dt = new DataTable();
        //通过 SqlDataAdapter 对象的 Fill 方法，将数据表信息添加到 DataSet 对象中
        da.Fill(dt);
        this.dataGridView1.DataSource = dt.DefaultView;      //用 dataGridView1 控件显示表信息
    }
    catch (Exception ey)
    {
        MessageBox.Show(ey.Message);
    }
}
```

通过选择 CheckBox 控件生成查询条件语句，每个 CheckBox 控件对应房源表中相关的字段。这里只列举一个字段的生成，其他相关字段生成可参见本书附带资源包中的源程序。实现代码如下：

```
private void checkBox1_CheckedChanged(object sender, EventArgs e)
{
    if (this.checkBox1.Checked)
    {
        txtName.Enabled = true;                                        //该控件可用
        if (strMidle == string.Empty)                                  //如果 strMidle 变量不为空
        {
            strMidle += "@"+"house_companyName" + "@";
        }
        else
        {
            strMidle += "house_companyName" + "@";
        }
        this.button1.Enabled = true; true;                             //该控件可用
    }
    else
    {
        txtName.Enabled = false; true;                                 //该控件不可用
    }
}
```

单击“查询”按钮，对 strMidle 变量进行相关处理，动态生成 SQL 语句。这里列出部分代码，其他可参见本书附带资源包中的源程序。

```
private void button1_Click(object sender, EventArgs e)
{
        //生成 where 条件字符串
        strSql.Append("select * from view_house where ");
```



```
if (strMidle.IndexOf("house_companyName")!=-1)//当在字符串中查找到指定字符时
{
    if (strWhere != string.Empty)
    {
        //设置模糊查询条件
        strWhere += "and " + "物业名称 like '%" + this.txtName.Text.Trim().ToString() + "%'" ;
    }
    else
    {
        strWhere += "物业名称 like '%" + this.txtName.Text.Trim().ToString() + "%'";
    }
    strMidle=strMidle.Replace("house_companyName", "#");
}
if (strMidle.IndexOf("huose_typeID")!= -1)
{
    if (strWhere != string.Empty)
    {
        strWhere += "and " + "类型='" + this.cboXing.Text.ToString() + "'" ;
    }
    else
    {
        strWhere += "类型='" + this.cboXing.Text.ToString() + "'" ;
    }
    strMidle=strMidle.Replace("huose_typeID", "#");
}
 …//其他代码可参见本书附带资源包
if (strMidle.IndexOf("house_ID") != -1)
{
    if (strWhere != string.Empty)
    {
      strWhere += "and " + "房屋编号 like '%" + this.textBox2.Text.Trim().ToString() + "%'";
    }
    else
    {
        strWhere += "房屋编号 like '%" + this.textBox2.Text.Trim().ToString() + "%'";
    }
    strMidle = strMidle.Replace("house_ID", "#");
}
try
{
    string strS = strWhere.Substring(strWhere.Length - 4);
    if (strS.Trim() == "and")
    {
        strWhere = strWhere.Substring(0, strWhere.Length - 4);//去掉尾 and
```

```
            }
        }
        catch { return; }
        strSql.Append(strWhere);
        string strK = strSql.ToString();
        try
        {
            //实例化 SqlDataAdapter 对象
            SqlDataAdapter da = new SqlDataAdapter(strK, con.conn);
            DataTable dt = new DataTable();                              //实例化 DataTable
            //通过 SqlDataAdapter 对象的 Fill 方法，将数据表信息添加到 DataSet 对象中
            da.Fill(dt);
            //dataGridView1 控件显示查找后的表信息
            this.dataGridView1.DataSource = dt.DefaultView;
            ChuShiHua();                                                 //调用自定义方法
            clearAll();                                                  //调用自定义方法
            this.button1.Enabled = false;
        }
        catch (Exception ey)
        {
            MessageBox.Show(ey.Message);
        }
        strWhere = string.Empty;
        strMidle = string.Empty;
        strSql.Remove(0,strSql.ToString().Length);
        button1.Enabled = false;
        this.textBox2.Text = "";
        this.textBox2.Enabled = false;
        checkBox11.Checked = false;
}
```

23.10 房源状态查询模块设计

23.10.1 房源状态查询模块概述

房源状态查询主要完成房源状态的查看，同时提供预订和取消预订的功能。房源状态以图标形式显示，灵活地运用了 ListView 控件的 View 属性。使用这种方式显示房源状态，为操作人员提供了更方便的查看方式，并且该模块还为客户提供了预约和取消预约房源的机会，从而留给客户更多的思考的空间，又一次体现出本系统人性化的设计思想。房屋状态查询窗体如图 23.25 所示。

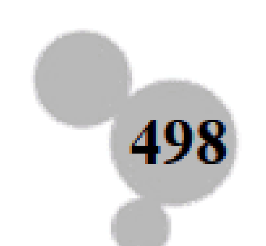




图 23.25　房源状态查询窗体

23.10.2　房源状态查询模块技术分析

在房源状态查询窗体中，使用 ListView 控件来显示房源的状态，对于不同状态的房源（如已租、未租、预订），ListView 控件会显示不同的图标，这样就使查看房源状态更方便。另外，ListView 控件还可以显示多种视图模式，这样就使得数据项的查看方式更加丰富，下面将介绍如何为 ListView 控件添加图标。

1．ListView 的图像列表属性和视图模式

ListView 控件有 3 个图像列表属性，分别是 LargeImageList、SmallImageList、StateImageList。List 视图模式、Details 视图模式、SmallIcon 视图模式将显示 SmallImageList 属性所指定的图像列表中的图像。LargeIcon 视图模式、Tile 视图将显示 LargeImageList 属性所指定的图像列表中的图像。列表视图还可以在大图标或小图标旁显示 StateImageList 属性中设置的一组附加图标。

2．通过编写代码为 ListView 控件添加图标

首先在 ImageList 控件中添加图标，然后将 ListView 控件的某个图像列表属性（如 SmallImageList 属性）设置为 ImageList 控件的实例，最后设置 ListView 控件的视图模式（如 View.Details 模式）。例如，下面的代码实现向 ListView 控件添加图标：

```
imageList1.Images.Add(Image.FromFile("01.png"));        //向 imageList1 中添加图标
imageList1.Images.Add(Image.FromFile("02.png"));        //向 imageList1 中添加图标
listView1.SmallImageList= imageList1;                   //设置控件的 SmallImageList 属性
listView1.View = View.Details;                          //设置视图模式
listView1.Items.Add("VB 项目整合");                      //向控件中添加项
listView1.Items.Add("C#项目整合");                       //向控件中添加项
listView1.Items[0].ImageIndex = 0;                      //控件中第 1 项的图标索引为 0
listView1.Items[1].ImageIndex = 1;                      //控件中第 2 项的图标索引为 1
```

3．通过属性窗口设置 ListView 控件的图标

除了通过编码可以实现为 ListView 控件添加图标外，还可以通过在 ListView 控件的属性窗口中设

置相关属性来实现添加图标，具体步骤如下：

（1）设置 ListView 控件的 View 属性为某种视图模式（Details、List、Tile 等）。

（2）根据上面设置的视图模式，将 ListView 控件的相应图像列表属性（SmallImageList、LargeImageList 或 StateImageList）设置为想要使用的现有 ImageList 控件。

（3）为每个具有关联图标的列表项设置 ImageIndex 属性或 StateImageIndex 属性，这个设置可以在“ListViewItem 集合编辑器”中进行（在 ListView 控件的“属性”窗口中，单击 Items 属性旁的省略号按钮，可以打开“ListViewItem 集合编辑器”）。

23.10.3 房源状态查询模块实现过程

房源状态查询模块的具体实现步骤如下。

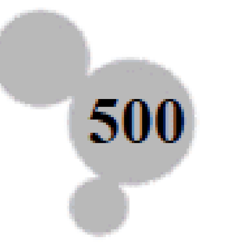
（1）新建一个 Windows 窗体，命名为 frmStateHouse.cs，用于查看房屋状态、预订和取消预订房屋。该窗体主要用到的控件及属性设置如表 23.12 所示。

表 23.12 房源状态查询窗体主要用到的控件

控件类型	控件 ID	主要属性设置	用途
TextBox	textBox1	将其 ReadOnly 属性设置为 false	手机号
RadioButton	rbHave	将 DropDownStyle 属性设置为 DropDownList	已租
	rbNone	同上	未租
	rbRemark	同上	预订
Button	button3	TextAlign 属性值有 9 种，这里设置为居中，即 Middle Center	预订
	button4	同上	取消预订
	button1	同上	查询
	button2	同上	显示全部
ListView	ListView1	将 ContextMenuStrip 属性设置为 cmLiftMothed	显示房源信息
ContextMenuStrip	cmLiftMothed	将 AutoClose 属性设置为 true	快捷菜单
ErrorProvider	epInfo	将 BlinkStyle 属性设置为 BlinkIfDifferentError	提示错误信息
ImageList	imgList	将 ImageSize 属性设置为“16,16”	绑定 ListView 控件以显示图标

（2）声明局部变量和公共类 ClsCon 的对象，通过 ClsCon 的对象调用类中的方法，实现数据库连接，代码如下：

```
public partial class frmStateHouse : Form
{
    string strSql = "select * from view_house";          //记录显示 view_house 表的 SQL 语句
    string strSqlWhereState = string.Empty;              //定义字符串变量
    string strThis = string.Empty;                       //定义字符串变量
    string strID = string.Empty;                         //定义字符串变量
    ClsCon con = new ClsCon();                           //实例化公共类 ClsCon
```



```
    public frmStateHouse()
    {
        InitializeComponent();
    }
    …//其他事件或方法的代码，可参见本书附带资源包
}
```

在 frmStateHouse 窗体的 Load 事件中，进行数据绑定，以显示房源状态相关信息。frmStateHouse 窗体的 Load 事件实现代码如下：

```
private void frmStateHouse_Load(object sender, EventArgs e)
{
    this.button1.Enabled = false;                          //禁用 button1 按钮
    ListInfo(strSql);                                      //显示所有房源信息
}
```

房屋中介系统提供了房屋 3 种状态的表现形式，即“未租”“预订”和“已租”，主要通过 ListInfo 方法显示房屋不同状态的图标。该功能的实现代码如下：

```
private void ListInfo(string SQL)                          //实现显示房屋不同状态下的图标
{
    con.ConDatabase();                                     //打开数据库的连接
    this.listView1.Items.Clear();                          //清空 listView1 控件
    SqlDataAdapter da = new SqlDataAdapter(SQL, con.conn); //实例化 SqlDataAdapter 对象
    DataTable dt = new DataTable();                        //实例化 DataTable 类
    da.Fill(dt);                                           //填充数据表实例
    foreach (DataRow dr in dt.Rows)                        //遍历数据表中的行信息
    {
        ListViewItem lv;                                   //实例化一个项
        if (dr[11].ToString() == "none")                   //如果该字段为空
        {
             lv= new ListViewItem(dr[0].ToString(), 0);
        }
        else if (dr[11].ToString() == "remark")
        {
            lv = new ListViewItem(dr[0].ToString(), 1);    //被预订状态
        }
        else
        {
            lv = new ListViewItem(dr[0].ToString(), 2);    //已租状态
        }
         //添加当前行中各字段的信息
        lv.SubItems.Add(dr[1].ToString());
        lv.SubItems.Add(dr[2].ToString());
        lv.SubItems.Add(dr[5].ToString());
        lv.SubItems.Add(dr[6].ToString());
        lv.SubItems.Add(dr[7].ToString());
```

```
            lv.SubItems.Add(dr[8].ToString());
            this.listView1.Items.Add(lv);
        }
        this.listView1.Columns[0].Width =120;                          //设置首列字段的宽度
}
```

设置房源显示模式的代码如下：

```
private void 平铺 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.listView1.View = View.LargeIcon;                              //平铺
}
private void 图标 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.listView1.View = View.SmallIcon;                              //图标
}
private void 列表 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.listView1.View = View.List;                                   //列表
}
private void 详细信息 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.listView1.View = View.Details;                                //详细信息
}
```

说明

要将 ImagList 控件和 ListView 控件绑定，需要将 ImagList 控件的 StateImageList、SmallImageList 和 LargeImageList 属性同 ListView 控件绑定，另外还要将 ShowGroups 属性设置为 true，这样才能达到想要的效果。

用户可以通过输入手机号码预订或取消预订房源信息，在 textBox1 控件中按下回车键时，判断用户是否有权享有这两项功能。该功能的实现代码如下：

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        try
        {
            ClsCon con = new ClsCon();                                 //实例化公共类 ClsCon
            con.ConDatabase();                                         //连接数据库
            SqlCommand cmd = new SqlCommand("select Max(user_names+'您的证件号
为:'+user_cardid) from tb_User where user_phone='" + textBox1.Text.Trim().ToString() +
"' and user_type<>'lend'", con.conn);                                  //通过加载 SQL 语句创建命令对象
            con.conn.Open();                                           //打开数据库的连接
```

```
            string strRe = cmd.ExecuteScalar().ToString();          //执行 SQL 语句
            con.closeCon();                                          //关闭数据库的连接
            if (strRe != "")                                         //如果该电话号码对应的求租人存在
            {
                //通过加载 SQL 语句创建命令对象，该 SQL 语句获取
                SqlCommand cmdl = new SqlCommand("select Max(house_ID) from tb_User where
                user_phone='" + textBox1.Text.Trim().ToString() + "'", con.conn);
                cmdl.Connection.Open();                              //打开数据库连接
                string strReS = cmdl.ExecuteScalar().ToString();     //获取最大房源编号
                con.closeCon();                                      //关闭连接
                if (strReS == "none")                                //若房子处于“未租”状态
                {
                    MessageBox.Show(strRe + "你可以预订房源");
                    this.button3.Enabled = true;                     //显示“预订”按钮
                    this.button4.Enabled = false;                    //隐藏“取消预订”按钮
                }
                else                                                 //若房子处于“已租”或“预订”状态
                {
                    this.button3.Enabled = false;                    //隐藏  “预订”按钮
                    this.button4.Enabled = true;                     //显示“取消预订”按钮
                }
                SendKeys.Send("{Tab}");
            }
            else                                                     //若该电话号码对应的求租人不存在
            {
                MessageBox.Show("电话号码不存在");                   //提示电话号码不存在
                this.textBox1.Select(0, this.textBox1.Text.Length);  //选中电话号码并获得焦点
                this.textBox1.Focus();
            }
        }
        catch (Exception ey)
        {
            MessageBox.Show(ey.Message);                             //弹出异常信息提示框
            con.conn.Close();                                        //关闭数据连接
        }
    }
}
```

说明

通过 Select 方法和 Focus 方法的并用，可以将所有信息选中并获得焦点，例如下面的代码：

```
this.textBox1.Select(0, this.textBox1.Text.Length)
this.textBox1.Focus ();
```

23.11　员工信息设置模块设计

23.11.1　员工信息设置模块概述

员工信息设置主要用于管理员工信息。例如给不同的员工分配系统的使用权限和工资等。当添加新员工时，通过触发器 trig_insetOfEmployeeinLogin 将其添加到系统用户表中，并且将密码及权限进行初始化。例如密码统一为 111，权限为普通员工。员工信息设置窗体如图 23.26 所示。

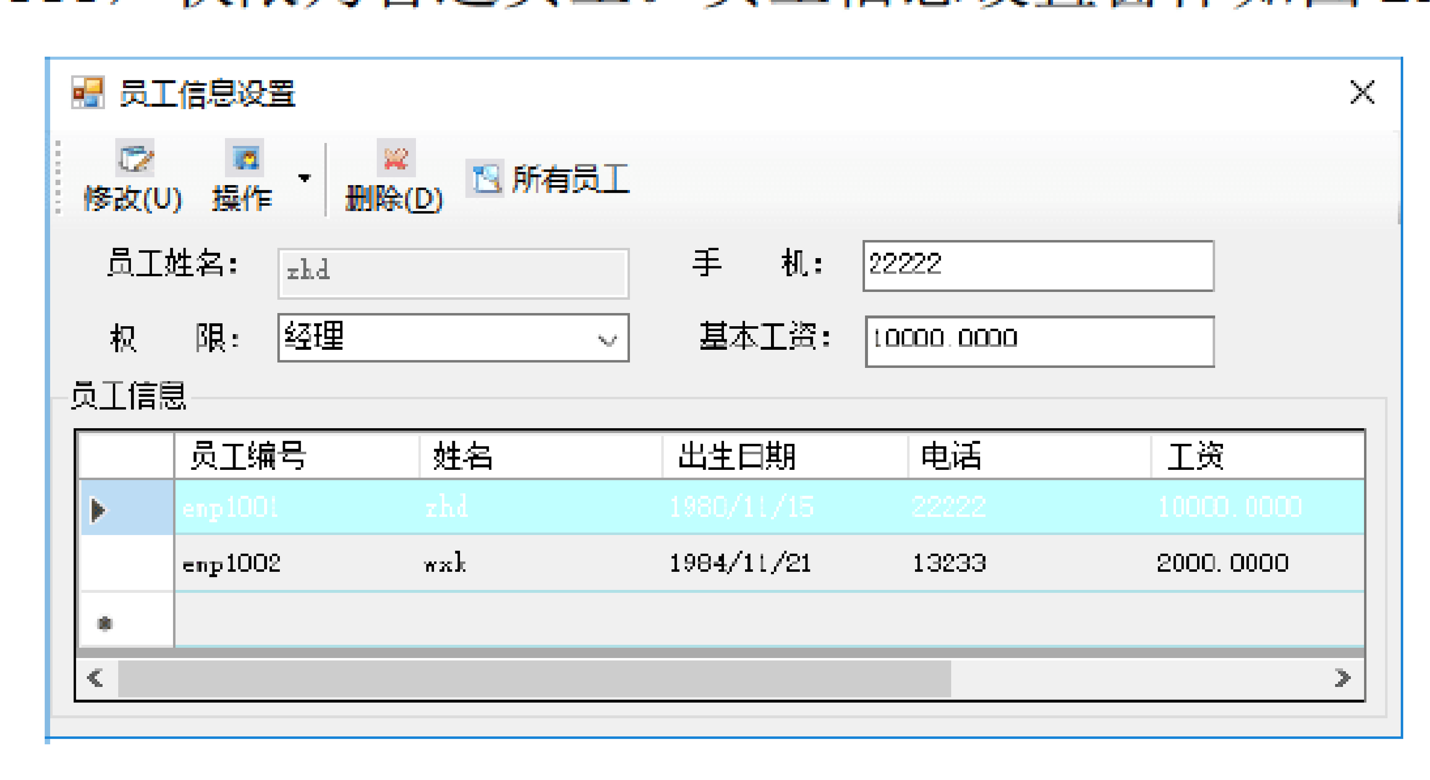
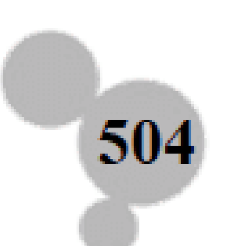


图 23.26　员工信息设置窗体

23.11.2　员工信息设置模块技术分析

本模块在实现时，用到了触发器，使用触发器可以自动将添加的新员工信息添加到系统用户表中，触发器是在 SQL Server 中依附于某个表编写的，代码中不用调用，它会自动执行。本模块中用到的触发器依附于 tb_employee 数据表，名称为 trig_insetOfEmployeeinLogin，代码如下：

```
CREATE TRIGGER [dbo].[trig_insetOfEmployeeinLogin]
ON
[dbo].[tb_employee]
for insert
AS
BEGIN
declare @lid varchar(10)
declare @led varchar(10)
declare @lna varchar(20)
declare @lpw varchar(15)
declare @lpo varchar(10)
select @lid=Max(login_id) from tb_login
if(@lid is null)
set @lid='log1001'
else
```



```
set @lid='log'+cast(substring(@lid,4,4)+1 as varchar(10))
select @led=employee_ID,@lna=employee_name from inserted
set @lpw='111'
set @lpo='0'
insert into tb_login values(@lid,@led,@lna,@lpw,@lpo)
end
```

23.11.3　员工信息设置模块实现过程

员工信息设置模块的具体实现步骤如下。

（1）新建一个 Windows 窗体，命名为 frmEmpleeyAll.cs，用于实现修改、删除和查看员工信息的功能，该窗体主要用到的控件及属性设置如表 23.13 所示。

表 23.13　员工信息设置窗体主要用到的控件

控件类型	控件名称	主要属性设置	用　途
TextBox	txtBasePay	将其 ReadOnly 属性设置为 false	基本工资
	txtName	同上	员工姓名
	txtPhone	同上	手机
ComboBox	cobPower	将其 DropDownStyle 属性设置为 DropDownList	权限列表
DataGridView	dataGridView	将 SelectionMode 属性设置为 FullRowSelect，以选取整行	显示员工信息
ToolStrip	toolStrip	将 TextDirection 属性设置为 Horizontal	控制操作

（2）声明局部变量及公共类 ClsCon 的对象，通过 ClsCon 的对象调用类中的方法，实现数据库连接，代码如下：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
using houseAgency.mothedCls;
namespace houseAgency
{
    public partial class frmEmpleeyAll : Form
    {
        ClsCon con = new ClsCon();              //实例化公共类 ClsCon
        string strTemp = string.Empty;          //定义字符串变量
        public frmEmpleeyAll()
        {
            InitializeComponent();
        }
```

```
        …//其他事件或方法的代码
    }
}
```

在 frmEmpleeyAll.cs 窗体的 Load 事件中，通过调用自定义 showAll 方法对 dataGridView 控件员工信息进行绑定。frmEmpleeyAll 窗体的 Load 事件关键代码如下：

```
private void frmEmpleeyAll_Load(object sender, EventArgs e)
{
    showAll();                              //自定义方法，显示 view_empleey 表中的所有信息
    this.cobPower.Items.Add("员工");        //在 ComboBox 控件中添加项
    this.cobPower.Items.Add("经理");
}
```

当用户单击 DataGridView 表格时，将表格中的员工信息显示在相应的文本框中，如图 23.26 所示，以上过程需要在 DataGridView 控件的 SelectionChanged 事件下完成。代码如下：

```
private void dataGridView1_SelectionChanged(object sender, EventArgs e)
{
    selectInfo();                           //调用自定义方法
}
```

自定义 selectInfo 方法，主要用来显示员工详细信息，代码如下：

```
private void selectInfo()
{
    try
    {
        string str = this.dataGridView1.SelectedCells[0].Value.ToString();
        SqlCommand cmd = new SqlCommand("select 姓名,电话,权限,工资 from view_empleey
        where 员工编号='" + str + "'", con.conn);
        cmd.Connection.Open();
        SqlDataReader dr = cmd.ExecuteReader();
        while (dr.Read())
        {
            txtName.Text = dr[0].ToString();
            txtPhone.Text = dr[1].ToString();
            txtBasePay.Text = dr[3].ToString();
            if (dr[2].ToString() == "0")
            {
                cobPower.Text = "员工";     //设置 ComboBox 控件的文本值
            }
            else
            {
                cobPower.Text = "经理";
            }
        }
        dr.Close();
        con.closeCon();
    }
```

```
        catch
        {}
}
```

单击“确定”按钮，通过视图和 INSTEAD OF 触发器并用，完成员工信息表和登录表的更新操作。代码如下：

```
private void tp_OK_Click(object sender, EventArgs e)
{
        string power = string.Empty;
        if (this.cobPower.Text == "员工")                    //如果权限是“员工”
        {
                power = "0";                                //设置该变量为“0”
        }
        else if (this.cobPower.Text == "经理")
        {
                power = "1";
        }
        if (strTemp == "Update")
        {
                float fmoney = Convert.ToSingle(this.txtBasePay.Text.Trim().ToString());
                SqlCommand cmd = new SqlCommand("update view_empleey set 权限='" + power + "',电
                话='" + this.txtPhone.Text.Trim().ToString() + "',工资='" +fmoney+ "' where 姓名='" +
                this.txtName.Text.Trim().ToString() + "'", con.conn);
                con.conn.Open();
                cmd.ExecuteNonQuery();                      //执行 SQL 的更新语句
                con.closeCon();
                showAll();
                MessageBox.Show("成功更改");
                strTemp = string.Empty;
        }
        else if( strTemp ==string.Empty)
        {
                MessageBox.Show("没有选取要对谁操作");
        }
}
```

23.12　小　结

优秀的应用系统软件需具备健壮性、灵活性以及良好的人性化界面。人性化可以让系统用户快速熟悉系统。本系统中的房源状态查询模块体现了这一特点，不同状态的房屋在浏览时显示出不同的图标，这样操作人员会对查询结果一目了然。同时为了方便数据的浏览，还提供多种房源状态查看方式。在此提醒读者应灵活运用每种控件，以方便用户的操作和使用。

第 24 章　客房管理系统
（C++ +SQL Server 2014 实现）

随着市场经济的发展，人们生活水平的不断提高及到异地办公、旅游人数的增多，宾馆、酒店业不断壮大，人们对住宿的要求也不断提高。传统的手工管理已经不能适应复杂的客房管理需求，各宾馆、酒店为了提高管理水平都先后使用计算机进行管理，这就需要开发出符合客房管理要求的管理系统。本章以软件工程的思想介绍了客房管理系统的开发过程。通过本章的学习，可以掌握以下要点：

☑　使用 SQL Server 2014 数据库

☑　使用 ADO 连接数据库

☑　通过 SQL 语句对数据库进行操作

24.1　开 发 背 景

随着我国市场经济的迅速发展，人们的生活水平有了显著提高，旅游经济和各种商务活动更促进了宾馆、酒店行业的快速发展。同时，随着宾馆、酒店的数量越来越多，人们的要求也越来越高，住宿行业的竞争愈演愈烈。如何在激烈的市场竞争中生存和发展，是每一个宾馆、酒店必须面临的问题。提高宾馆、酒店的经营管理，为顾客提供更优质的服务，同时降低运营成本是发展的关键。面对信息时代的机遇和挑战，利用科技手段提高企业管理效率无疑是一条行之有效的途径。计算机的智能化管理技术可以极大限度地提高服务管理水平，进行准确、快捷和高效的管理。因此，采用全新的计算机客房管理系统，已成为提高宾馆、酒店管理效率，改善服务水平的重要手段之一。管理方面的信息化已成为现代化管理的重要标志。

以往的人工操作管理中存在着许多问题，例如：

☑　人工计算账单容易出现错误。

☑　收银工作中容易账单丢失。

☑　客人具体消费信息难以查询。

☑　无法对以往营业数据进行查询。

24.2　需 求 分 析

根据宾馆、酒店客房的具体情况，系统主要功能应该包括：

☑　住宿管理。
☑　客房管理。
☑　挂账管理。
☑　查询统计。
☑　日结。
☑　系统设置。

24.3　系 统 设 计

24.3.1　系统目标

面对宾馆、酒店行业的高速发展和行业信息化发展的过程中出现的各种情况，客房管理系统应能够达到以下目标：

☑　实现多点操作的信息共享，相互之间的信息传递准确、快捷和顺畅。
☑　服务管理信息化，可随时掌握客人住宿、挂账率、客房状态等情况。
☑　系统界面友好美观，操作简单易行，查询灵活方便，数据存储安全。
☑　客户档案、挂账信息和预警系统相结合，可对往来客户进行住宿监控，防止坏账的发生。
☑　通过客房管理系统的实施，可逐步提高宾馆、酒店客房的管理水平，提升员工的素质。
☑　系统维护方便可靠，有较高的安全性，满足实用性、先进性的要求。

24.3.2　系统功能结构

根据宾馆、酒店客房的具体情况，系统主要功能包括以下几个方面。

☑　住宿管理：客房预订、调房登记、住宿登记、追加押金和退宿结账。
☑　客房管理：客房设置、宿费提醒和房态查看。
☑　挂账管理：挂账查询和客户结款。
☑　查询统计：预订房查询、住宿查询、退宿查询和客房查询。
☑　日结：登记预收报表、客房销售报表和客房销售统计。
☑　系统设置：系统初始化、密码设置、权限设置和操作员设置。

为了清晰、全面地介绍客房管理系统的功能，以及各个模块间的从属关系，下面以结构图的形式

给出系统功能，如图 24.1 所示。

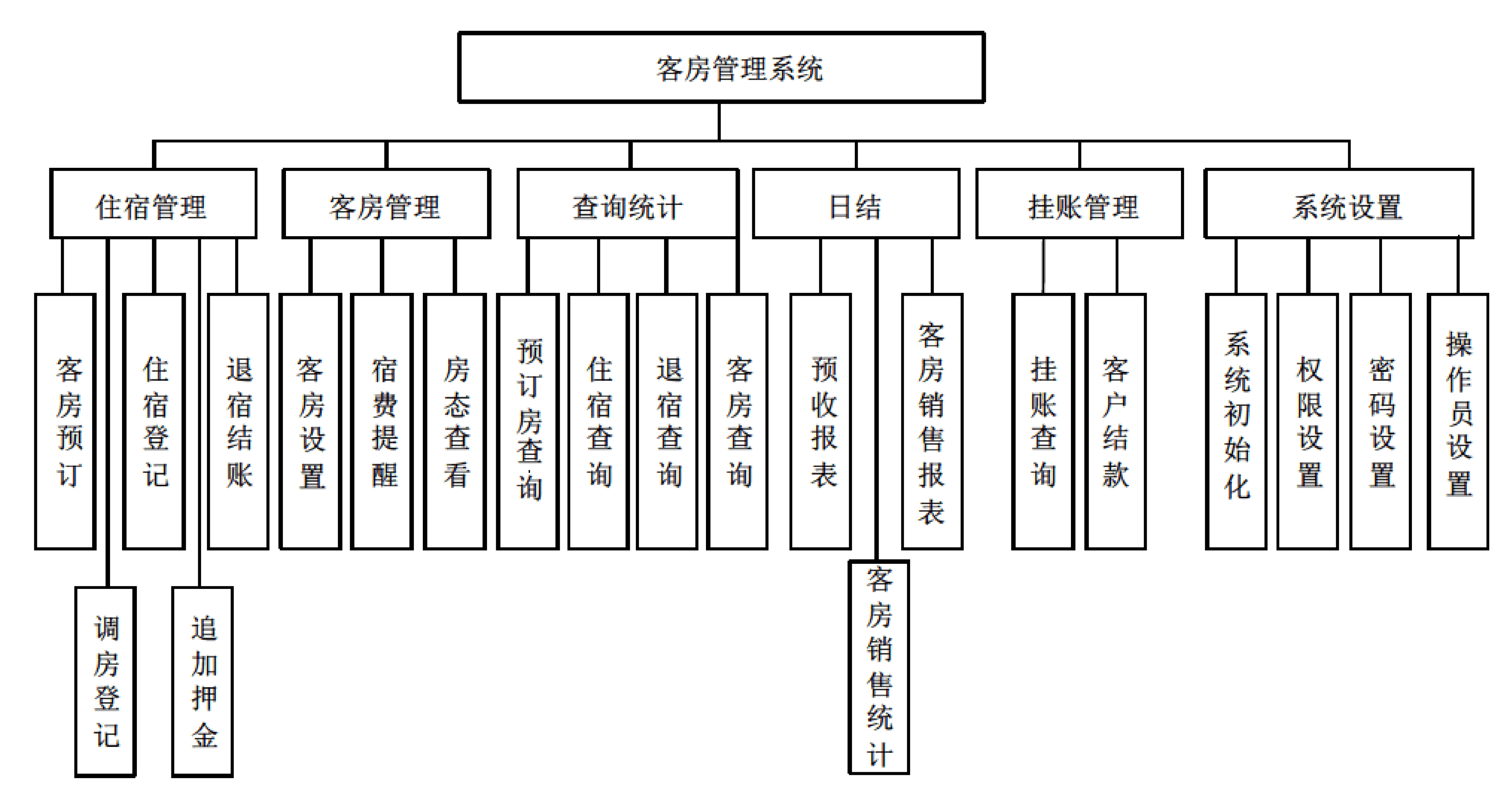


图 24.1　客房管理系统的功能结构图

24.3.3　系统预览

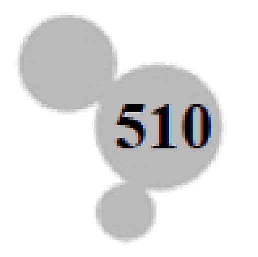
本系统包含多个功能模块，这里给出主要的窗体界面图，帮助大家更快地了解本系统的结构功能。

主窗体包含打开其他窗体的菜单和主要功能的命令按钮，是程序最主要的界面。其运行效果如图 24.2 所示。



图 24.2　系统主界面

客房预订模块主要用来记录客户的预订客房信息，实现对预订信息的管理。其界面效果如图 24.3 所示。



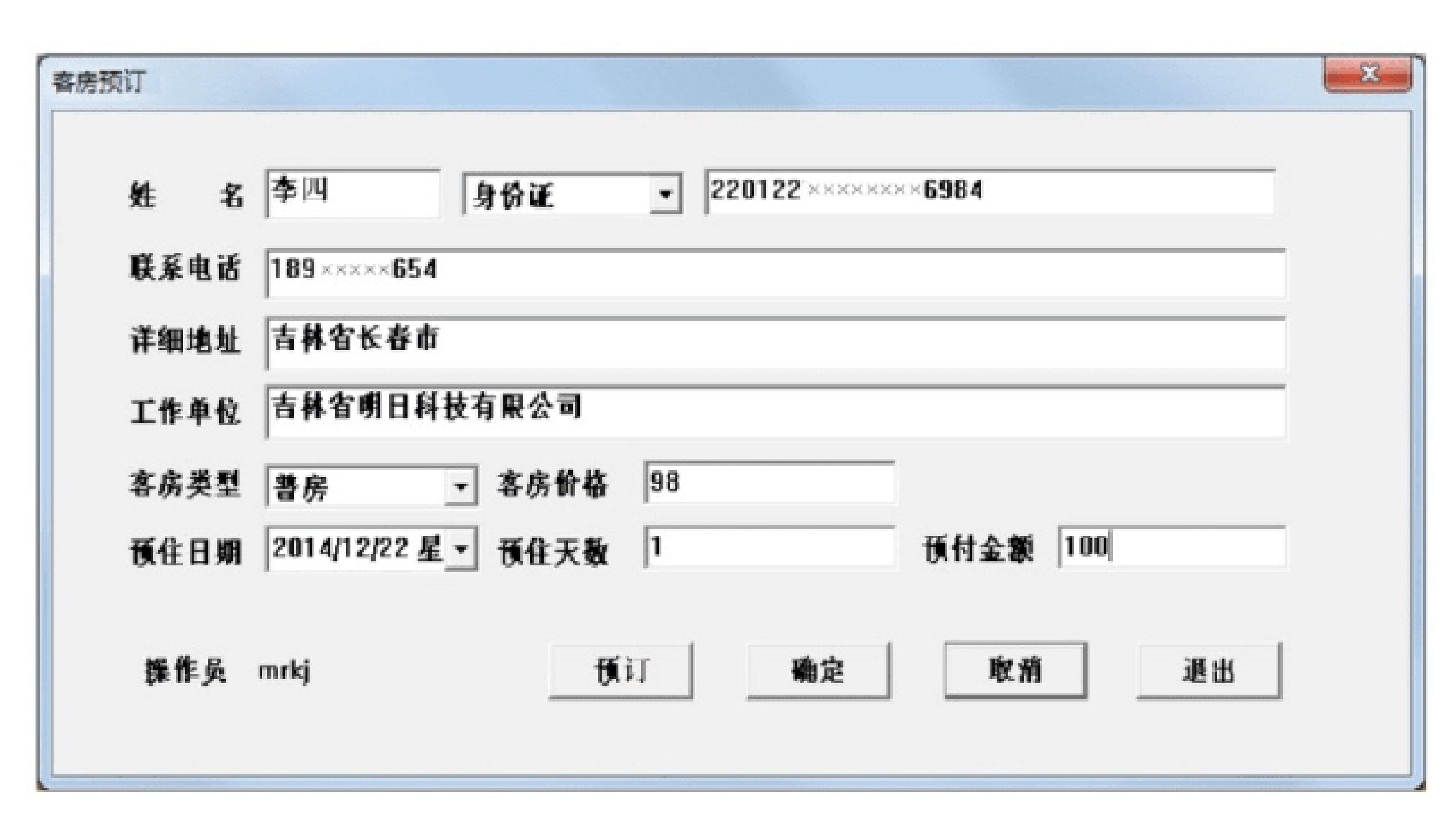


图 24.3　客房预订界面

追加押金模块主要用来实现记录追加押金的信息，并显示客人的当前住宿信息。其运行界面如图 24.4 所示。



图 24.4　追加押金界面

调房登记模块主要用来实现记录客人调房信息。其运行界面如图 24.5 所示。



图 24.5　调房登记界面

24.3.4　业务流程图

客房管理系统的业务流程图如图 24.6 所示。



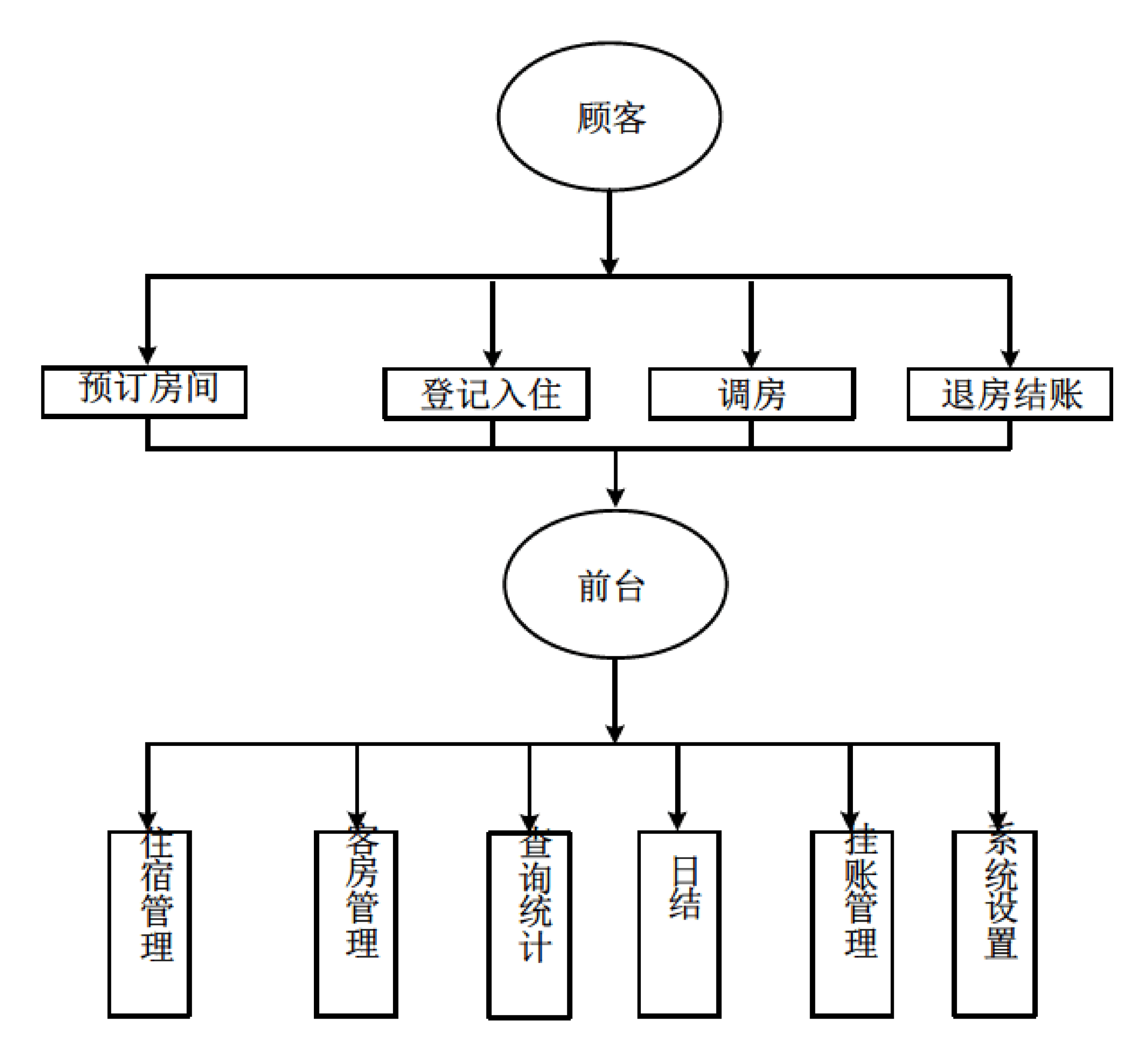


图 24.6　客房管理系统的业务流程图

24.3.5　数据库设计

1．数据库概要说明

在 SQL Server 2014 数据库中建立名为 myhotel 的数据库，设计 checkinregtable、checkoutregtable、guazhanginfo、kfyd、regmoneytable、roomsetting、setability 和 usertalbe 数据表。

图 24.7 所示即为本系统数据库中的数据表结构图，该结构图中包含系统所有的数据表，可以清晰地反映数据库信息。



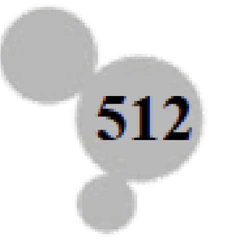
名称	架构
系统表	
checkinregtable	dbo
checkoutregtable	dbo
guazhanginfo	dbo
kfyd	dbo
regmoneytable	dbo
roomsetting	dbo
setability	dbo
usertalbe	dbo

图 24.7　数据库概要说明

2．主要数据表结构

下面给出主要数据表的结构，其他表的结构参见数据库。

☑ 住宿登记表（checkinregtable）：主要用于记录住宿登记信息，包括住宿人信息、房间信息和住宿情况，该表结构如图 24.8 所示。

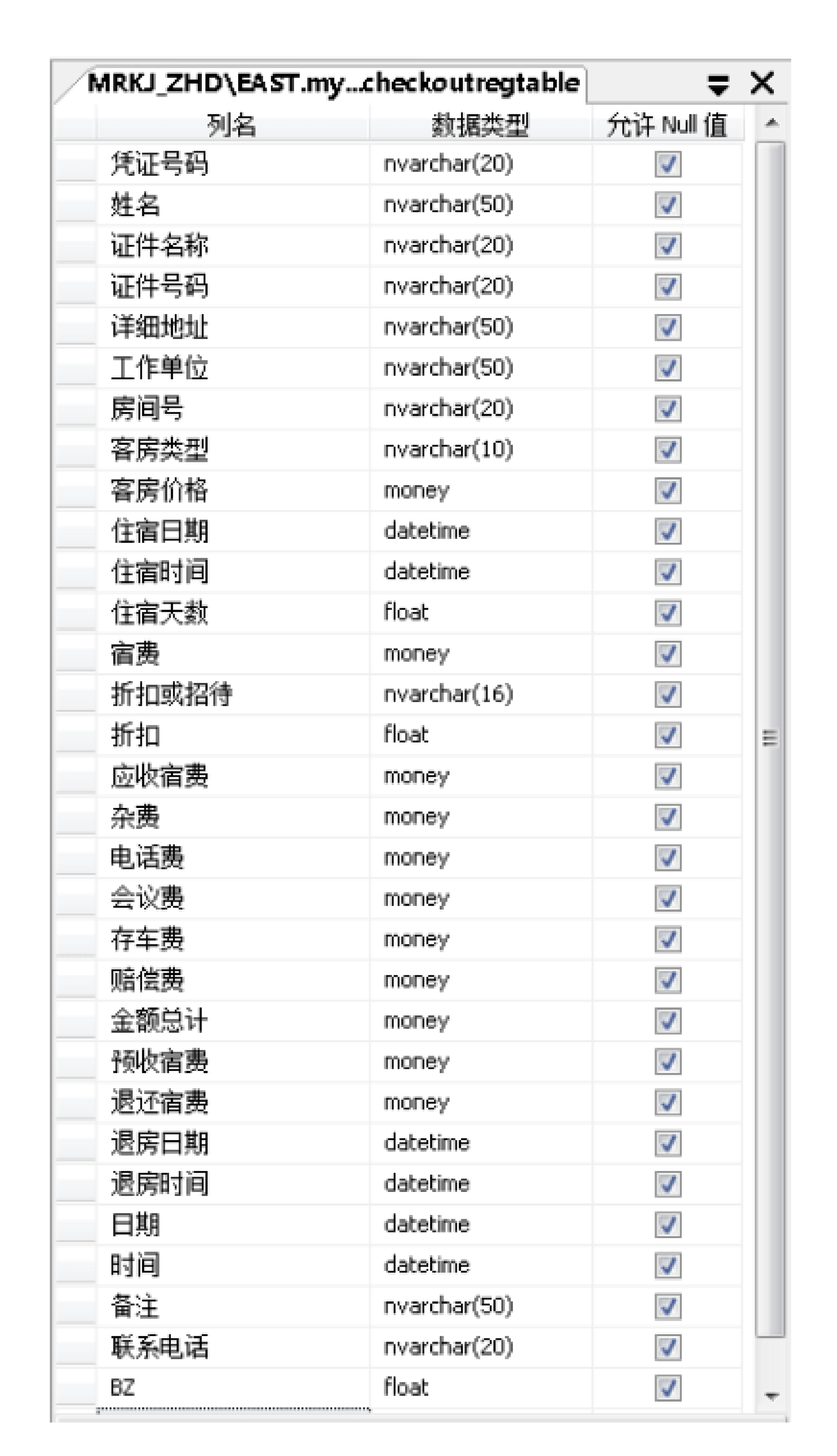




MRKJ_ZHD\EAST.my...o.checkinregtable

列名	数据类型	允许 Null 值
凭证号码	nvarchar(20)	✓
姓名	nvarchar(50)	✓
证件名称	nvarchar(20)	✓
证件号码	nvarchar(20)	✓
详细地址	nvarchar(50)	✓
出差事由	nvarchar(50)	✓
房间号	nvarchar(20)	✓
客房类型	nvarchar(10)	✓
联系电话	nvarchar(20)	✓
客房价格	money	✓
住宿日期	datetime	✓
住宿时间	datetime	✓
住宿天数	float	✓
宿费	money	✓
折扣	float	✓
应收宿费	money	✓
预收金额	money	✓
提醒日期	datetime	✓
退宿日期	datetime	✓
备注	nvarchar(50)	✓
标志	nvarchar(1)	✓
日期	datetime	✓
时间	datetime	✓
结款方式	nvarchar(10)	✓
退宿时间	datetime	✓
提醒时间	datetime	✓
摘要	nvarchar(200)	✓
BZ	float	✓

图 24.8　住宿登记表



MRKJ_ZHD\EAST.my...checkoutregtable

列名	数据类型	允许 Null 值
凭证号码	nvarchar(20)	✓
姓名	nvarchar(50)	✓
证件名称	nvarchar(20)	✓
证件号码	nvarchar(20)	✓
详细地址	nvarchar(50)	✓
工作单位	nvarchar(50)	✓
房间号	nvarchar(20)	✓
客房类型	nvarchar(10)	✓
客房价格	money	✓
住宿日期	datetime	✓
住宿时间	datetime	✓
住宿天数	float	✓
宿费	money	✓
折扣或招待	nvarchar(16)	✓
折扣	float	✓
应收宿费	money	✓
杂费	money	✓
电话费	money	✓
会议费	money	✓
存车费	money	✓
赔偿费	money	✓
金额总计	money	✓
预收宿费	money	✓
退还宿费	money	✓
退房日期	datetime	✓
退房时间	datetime	✓
日期	datetime	✓
时间	datetime	✓
备注	nvarchar(50)	✓
联系电话	nvarchar(20)	✓
BZ	float	✓

图 24.9　退宿登记表

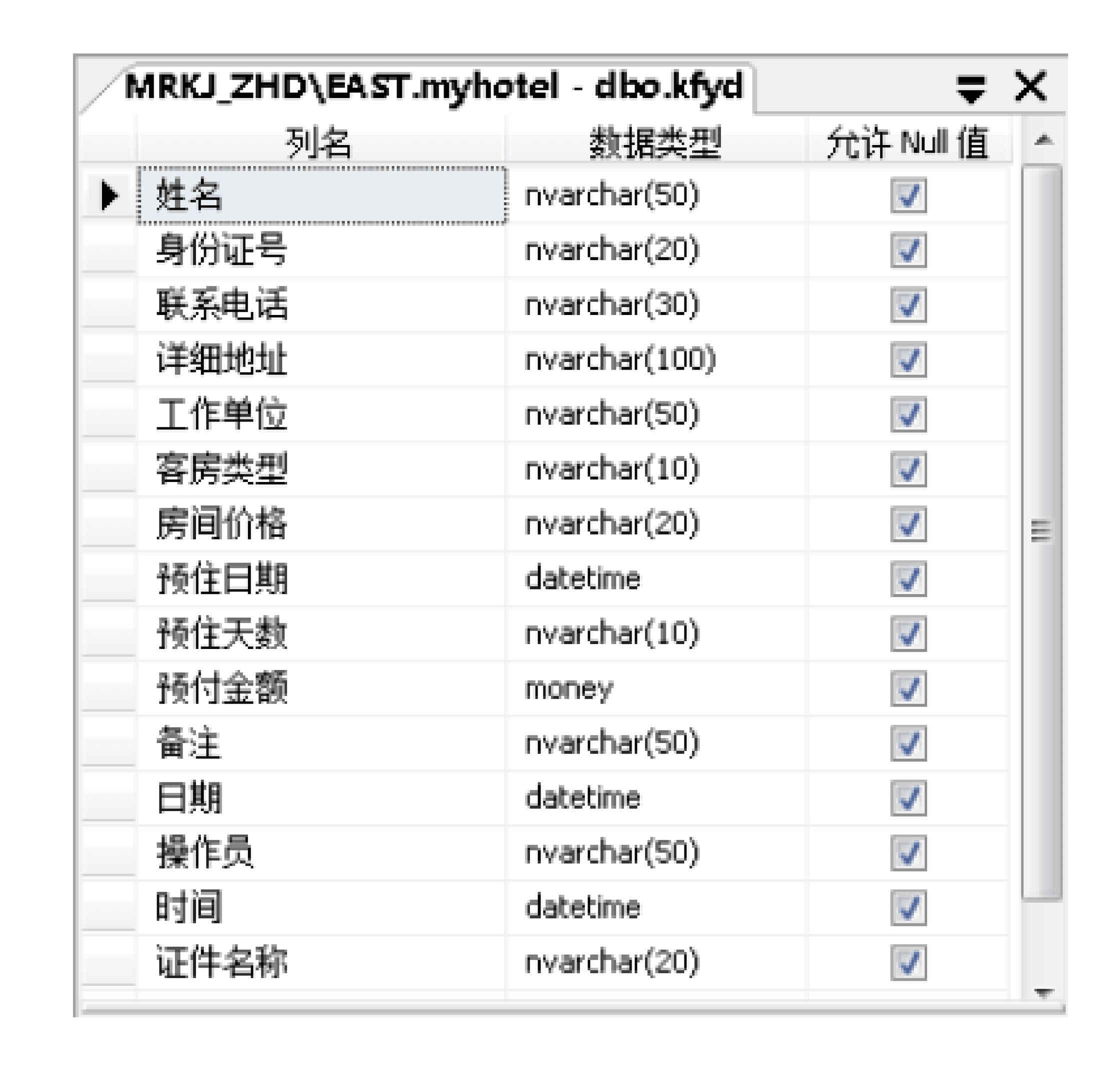
☑　退宿登记表（checkoutregtable）：主要用于记录退房登记信息，包括住宿和退房情况等信息，该表结构如图 24.9 所示。

☑　客房设置表（roomsetting）：用于存储客房的基本信息和客房状态等信息，该表结构如图 24.10 所示。

MRKJ_ZHD\EAST.my...- dbo.roomsetting

列名	数据类型	允许 Null 值
房间号	nvarchar(30)	☐
房间类型	nvarchar(20)	☐
价格	money	☐
房态	nvarchar(8)	✓
标志	bit	✓
备注	nvarchar(100)	✓
配置	nvarchar(100)	✓
使用设置	nvarchar(10)	✓
营业日期	datetime	✓

图 24.10　客房设置表



MRKJ_ZHD\EAST.myhotel - dbo.kfyd

列名	数据类型	允许 Null 值
姓名	nvarchar(50)	✓
身份证号	nvarchar(20)	✓
联系电话	nvarchar(30)	✓
详细地址	nvarchar(100)	✓
工作单位	nvarchar(50)	✓
客房类型	nvarchar(10)	✓
房间价格	nvarchar(20)	✓
预住日期	datetime	✓
预住天数	nvarchar(10)	✓
预付金额	money	✓
备注	nvarchar(50)	✓
日期	datetime	✓
操作员	nvarchar(50)	✓
时间	datetime	✓
证件名称	nvarchar(20)	✓

图 24.11　客房预订表

☑　客房预订表（kfyd）：用于记录客房预订信息，包括预订人信息和房间信息等，该表结构如图 24.11 所示。

24.4　主窗体设计

24.4.1　主窗体概述

主窗体界面是应用程序提供给用户访问其他功能模块的平台，根据实际需要，客房管理系统的主界面采用了传统的“菜单/工具栏/状态栏”风格，如图 24.12 所示。



图 24.12　系统主界面

24.4.2　主窗体实现过程

1．客户区设计

在生成的对话框内添加图片、静态文本、标签、编辑框和按钮等资源。

主要控件的 ID 和属性如表 24.1 所示。

表 24.1　主要控件的 ID 和属性

控件 ID	标　　题	控件 ID	标　　题
ID_BTN_borrowroom	开房	ID_BTN_daysummery	日结
ID_BTN_returnroom	结账	ID_BTN_alert	提醒
ID_BTN_mainfind	查询	ID_CLOSE	退出

2．菜单设计

（1）选择 Insert/Resource 命令，打开插入资源对话框，如图 24.13 所示。

（2）选择 Menu 选项，单击新建按钮，插入空白菜单，设置 ID 属性为 IDR_mainMENU，然后按

照图 24.14 所示的界面编辑菜单项。



图 24.13　插入资源对话框



图 24.14　编辑菜单项

主菜单的各个子菜单的 ID 和标题属性如表 24.2 所示。

表 24.2　各个子菜单的 ID 和标题属性

控件 ID	标　题	控件 ID	标　题
ID_MENU_checkinreg	住宿登记	ID_MENU_regmoneytable	登记预收报表
ID_MENU_roomsetting	客房设置	ID_MENU_saleroomtable	客房销售报表
ID_MENU_checkout	退宿结账	ID_MENU_saleroomsummary	客房销售统计
ID_MENU_addmoney	追加押金	ID_MENU_adm_setting	操作员设置
ID_MENU_changeroomreg	调房登记	ID_MENU_pwd_setting	密码设置
ID_MENU_findroom	客房查询	ID_MENU_setting_begin	初始化
ID_MENU_findguazhang	挂账查询	ID_MENU_setting_ability	权限设置
ID_MENU_guazhangmoney	客户结款	ID_MENU_findroomstate	房态查看
ID_MENU_findcheckinreg	住宿查询	ID_MENU_roomprebook	客房预订
ID_MENU_findcheckoutreg	退宿查询	ID_MENU_findprebookroom	预订房查询
ID_MENU_findroomfee	宿费提醒		

3．代码分析

（1）系统主界面操作可以根据用户的权限设定，所以应加入连接数据库功能，故在 stdafx.h 文件中加入以下代码，提供加入 ADO 的支持。

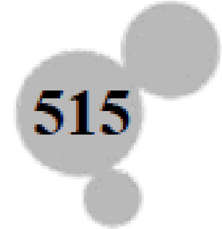
```
//添加 ADO 支持
#import "c:\program files\common files\system\ado\msado15.dll" \ no_namespace \ rename ("EOF", "adoEOF")
```

并在 Myhotel.h 中加入以下代码：

```
CDatabase m_DB;
_ConnectionPtr m_pConnection;
```

此外，在 myhotel.cpp 的初始化函数中加入连接数据库的代码：

```
try                                                              //连接数据库
```



```
{
 CString strConnect;
 strConnect.Format("DSN=myhotel;");
 if(!m_DB.OpenEx(strConnect,CDatabase::useCursorLib))
 {
 AfxMessageBox("Unable to Connect to the Specified Data Source");
 return FALSE ;
 }
}
catch(CDBException *pE)                                     //抛出异常
{
     pE->ReportError();
     pE->Delete();
     return FALSE;
}
//初始化 COM，创建 ADO 连接等操作
AfxOleInit();
m_pConnection.CreateInstance(__uuidof(Connection));
//在 ADO 操作中建议语句中要常用 try...catch()来捕获错误信息
try
{
     //打开本地数据库
     m_pConnection->Open("Provider=MSDASQL.1;Persist Security Info=False;Data Source = myhotel","","",
adModeUnknown);
}
catch(_com_error e)                                              //抛出可能发生的异常
{
     AfxMessageBox("数据库连接失败，确认数据库配置正确!");
     return FALSE;
}
```

（2）主窗口初始化时，需要根据登录操作员的权限来设置其可以进行的操作，此功能由函数 setuserability 来完成，代码如下：

```
void CMyhotelDlg::setuserability()
{
     m_pRecordset.CreateInstance(__uuidof(Recordset));
     _variant_t var,varIndex;

     //loguserid="操作员 01";
     CString strsqlshow;
     strsqlshow.Format("SELECT * FROM setability where  操作员='%s'",loguserid);

     try                                                           //打开数据库连接
     {
          m_pRecordset->Open((_variant_t)(strsqlshow),                //查询表中所有字段
               theApp.m_pConnection.GetInterfacePtr(),                //获取库接库的 IDispatch 指针
                                   adOpenDynamic,
                                   adLockOptimistic,
                                   adCmdText);
```

```
}
catch(_com_error *e)                                    //捕获异常的发生
{
    AfxMessageBox(e->ErrorMessage());
}
mynenu=AfxGetMainWnd()->GetMenu();                      //获得主菜单指针
CString ling="0";
try
{
    if(!m_pRecordset->BOF)                              //判断指针是否在数据集最后
        m_pRecordset->MoveFirst();
    else
    {
        AfxMessageBox("表内数据为空");
        return;
    }
     MessageBox("eeeeeeeeee");
     //读取数据表内客房预订字段内容
     var = m_pRecordset->GetCollect("客房预订");
     if(var.vt != VT_NULL)
     {
       if((LPCSTR)_bstr_t(var)==ling)                   //判断是否有权限操作客房预订模块
       {                                                //如果没有权限就使该菜单呈灰色显示
       EnableMenuItem(mynenu->m_hMenu,ID_MENU_roomprebook,MF_DISABLED|MF_GRAYED);
       }

    }
    //读取数据表内住宿登记字段内容
    var = m_pRecordset->GetCollect("住宿登记");
    if(var.vt != VT_NULL)
    {
       if((LPCSTR)_bstr_t(var)==ling)                   //判断是否有权限操作住宿登记模块
       {                                                //如果没有权利就使该菜单呈灰色显示
           EnableMenuItem(mynenu->m_hMenu,ID_MENU_checkinreg,MF_DISABLED|MF_GRAYED);
       }
    }
    //读取数据表内追加押金字段内容
    var = m_pRecordset->GetCollect("追加押金");
    if(var.vt != VT_NULL)
    {
        if((LPCSTR)_bstr_t(var)==ling)                  //判断是否有权限操作追加押金模块
        {                                               //如果没有权利就使该菜单呈灰色显示
            EnableMenuItem(mynenu->m_hMenu,ID_MENU_addmoney,MF_DISABLED|MF_GRAYED);
        }

    }
    //读取数据表内调房登记字段内容
    var = m_pRecordset->GetCollect("调房登记");
    if(var.vt != VT_NULL)
```

```
        {
            if((LPCSTR)_bstr_t(var)==ling)                  //判断是否有权限操作调房登记模块
             {                                              //如果没有权利就使该菜单呈灰色显示
                EnableMenuItem(mynenu->m_hMenu,ID_MENU_changeroomreg,
                               MF_DISABLED |MF_GRAYED);
             }

        }

       …//其他菜单设计代码参见本书附带资源包
        mynenu->Detach();
        DrawMenuBar();                                      //重绘主菜单
    catch(_com_error *e)                                    //捕获异常
    {
        AfxMessageBox(e->ErrorMessage());                   //弹出错误信息框
    }
    m_pRecordset->Close();                                  //关闭记录集
    m_pRecordset = NULL;
}
```

（3）在实现主窗体时，需要创建几个函数。创建 OnSysCommand 函数，代码如下：

```
void CMyhotelDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
     if ((nID & 0xFFF0) == IDM_ABOUTBOX)
     {
          CAboutDlg dlgAbout;
          dlgAbout.DoModal();
     }
     else
     {
          CDialog::OnSysCommand(nID, lParam);
     }
}
```

创建 OnPaint 函数，代码如下：

```
void CMyhotelDlg::OnPaint()
{CPaintDC dc(this); // device context for painting
     CBitmap bit;
     CDC memDC;
     CRect rect;
     this->GetClientRect(&rect);
     bit.LoadBitmap(IDB_MAINBK);
     BITMAP bmpInfo;
     bit.GetBitmap(&bmpInfo);
     int imgWidth = bmpInfo.bmWidth;
     int imgHeight = bmpInfo.bmHeight;
     memDC.CreateCompatibleDC(&dc);
     memDC.SelectObject(&bit);
     dc.StretchBlt(0,0,rect.Width(),rect.Height(),&memDC,0,0,imgWidth,imgHeight,SRCCOPY);
```

```
    memDC.DeleteDC();
    bit.DeleteObject();

}
```

创建 OnQueryDragIcon、OnMENUcheckinreg、OnBTNborrowroom 函数，代码如下：

```
HCURSOR CMyhotelDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

void CMyhotelDlg::OnMENUcheckinreg()
{
    // TODO: Add your command handler code here
    CCheckinregdlg mycheckindlg;
   mycheckindlg.DoModal();
}

void CMyhotelDlg::OnBTNborrowroom()
{
    // TODO: Add your control notification handler code here
    OnMENUcheckinreg();
}
```

创建 OnMENUroomsetting、OnMENUcheckout、OnBTNreturnroom 函数，代码如下：

```
void CMyhotelDlg::OnMENUroomsetting()
{
    // TODO: Add your command handler code here
    CSetroomdlg mysetroomdlg;
    mysetroomdlg.DoModal();
}
void CMyhotelDlg::OnMENUcheckout()
{
    // TODO: Add your command handler code here
    CCheckoutdlg mycheckoutdlg;
   mycheckoutdlg.DoModal();
}
void CMyhotelDlg::OnBTNreturnroom()
{
    // TODO: Add your control notification handler code here
    OnMENUcheckout();
}
```

创建 OnMENUaddmoney、OnMENUchangeroomreg、OnMENUfindroom 函数，代码如下：

```
void CMyhotelDlg::OnMENUaddmoney()
```

```
{
    // TODO: Add your command handler code here
    CAddmoneydlg myaddmoneydlg;
   myaddmoneydlg.DoModal();
}

void CMyhotelDlg::OnMENUchangeroomreg()
{
    // TODO: Add your command handler code here
    CChangeroomdlg mychangeroomdlg;
   mychangeroomdlg.DoModal();
}

void CMyhotelDlg::OnMENUfindroom()
{
    // TODO: Add your command handler code here
    CFindroomdlg myfindroomdlg;
   myfindroomdlg.DoModal();
}
```

24.5　登录模块设计

24.5.1　登录模块概述

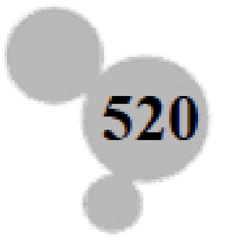
为了防止非法用户进入系统，本软件设计了系统登录窗口。在程序启动时，首先弹出“登录”窗口，要求用户输入登录信息，如果用户输入不合法，将禁止进入系统。登录模块的运行效果如图 24.15 所示。



图 24.15　登录模块的运行效果

24.5.2　登录模块技术分析

本模块使用 CUserset 类实现对数据源的连接。这里是通过 ODBC 数据源进行连接的，在连接数据



库之前，要先在系统上创建一个名为 myhotel 的数据源。userset.cpp 中的代码如下：

```
CString CUserset::GetDefaultConnect()
{
	return _T("ODBC;DSN=myhotel");
}
CString CUserset::GetDefaultSQL()
{
	return _T("[dbo].[user]");
}
```

24.5.3　登录模块设计过程

（1）选择 Insert/Resource 命令，打开插入资源对话框。选择 Dialog 选项，单击新建按钮，插入新的对话框。

（2）利用类向导为此对话框资源设置属性。在 Name 文本框中输入对话框类名，如 CLoginDlg，在 Base class 下拉列表框中选择一个基类，这里为 CDialog，单击确定按钮创建对话框。

（3）在工作区的资源视图中选择新创建的对话框，向对话框中添加静态文本、下拉列表框、编辑框和按钮等资源。主要控件的 ID 和属性如表 24.3 所示。

表 24.3　主要控件的 ID 和属性

控件 ID	对应变量/标题属性	控件 ID	对应变量/标题属性
IDC_COMBO_username	m_username	IDOK	确定
IDC_password	m_password	IDCANCEL	取消

（4）建立和数据库的映射，利用类向导建立记录集的映射类，如图 24.16 所示。

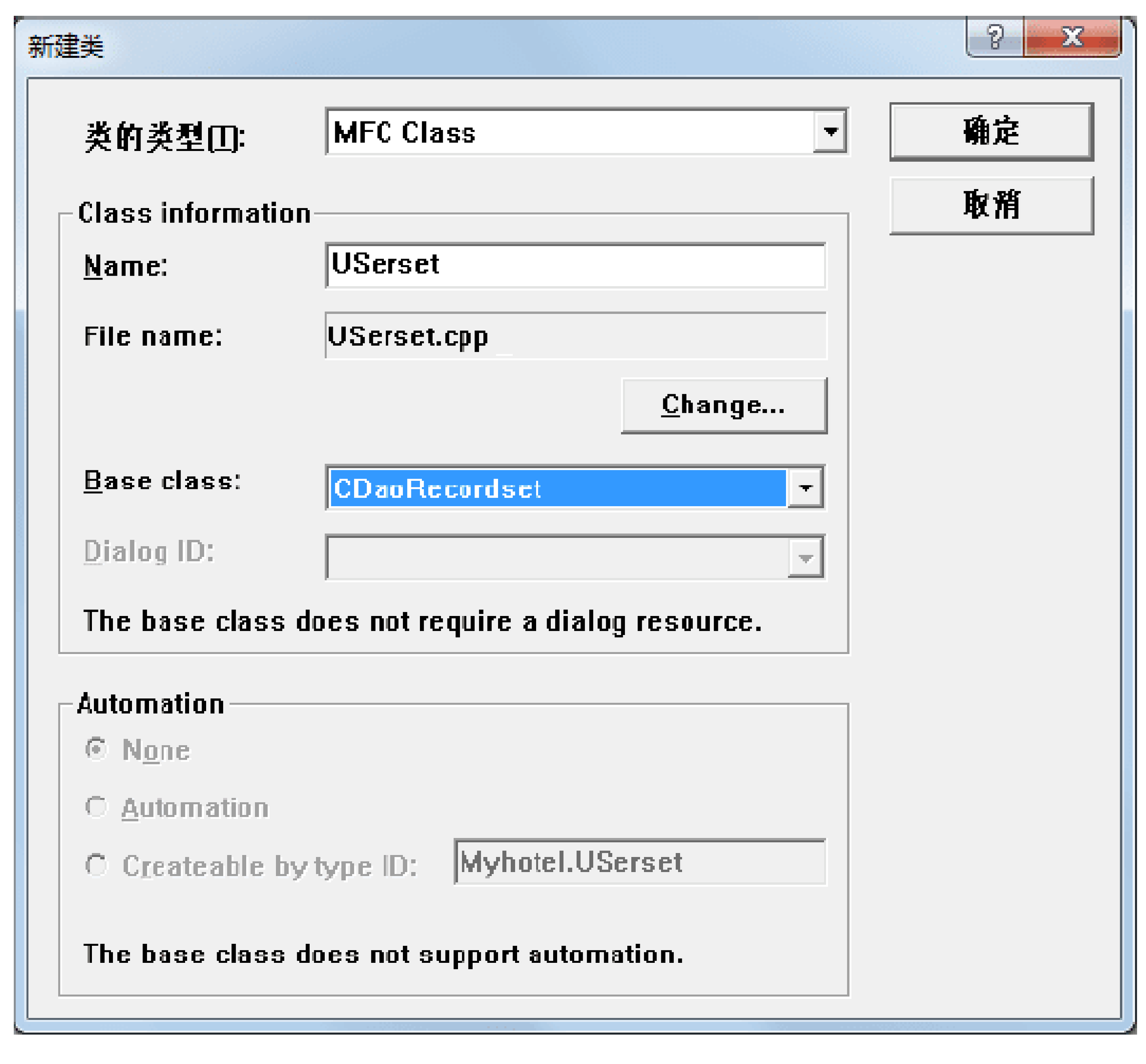


图 24.16　新建类对话框

选择 Base class 为 CDaoRecordset，单击确定按钮进入下一步，如图 24.17 所示。

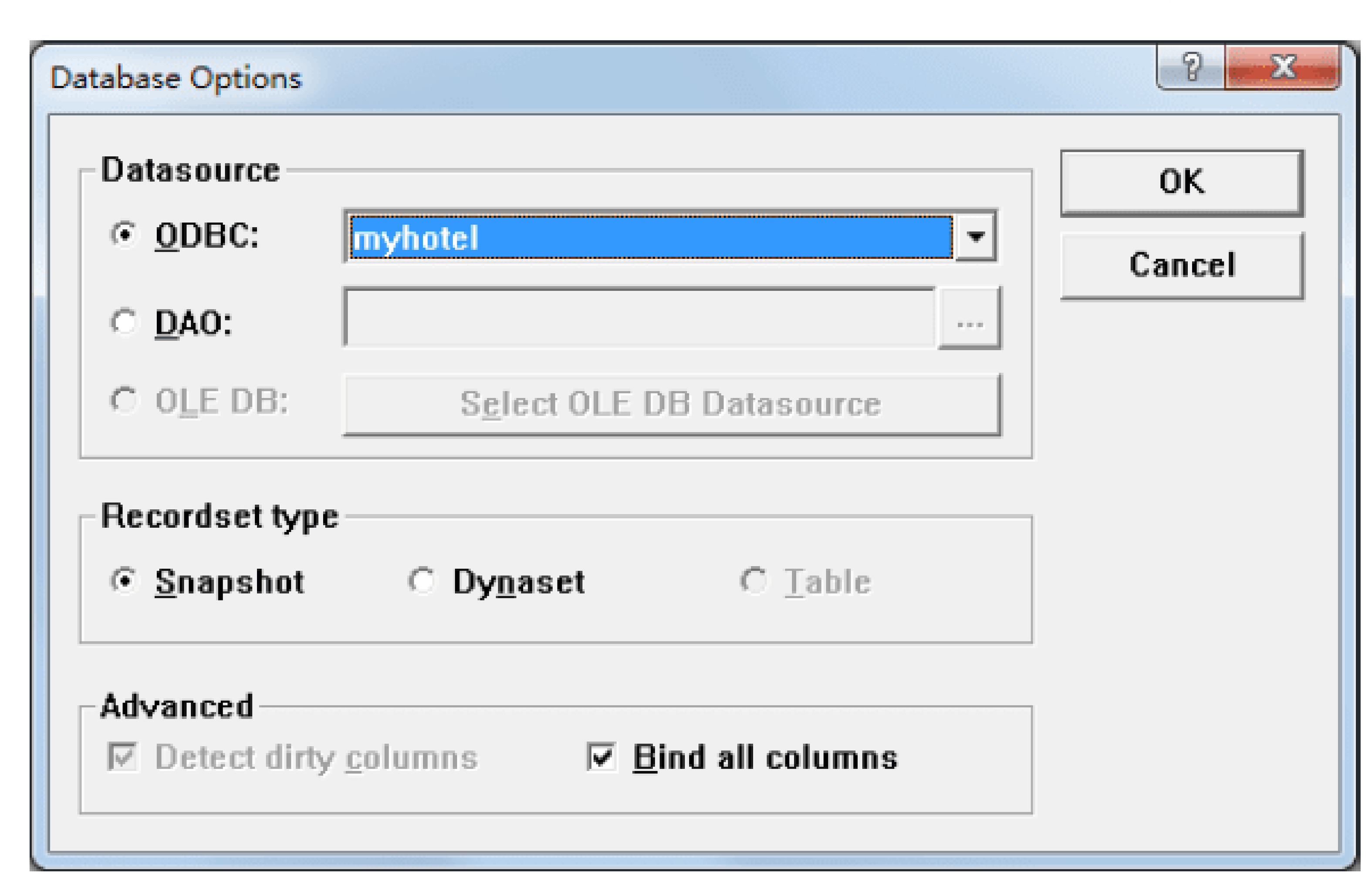


图 24.17　Database Options 对话框

选择数据源类型为 ODBC，并选择所使用的数据源，此处选择 myhotel 数据源，单击 OK 按钮，进入下一步，如图 24.18 所示。

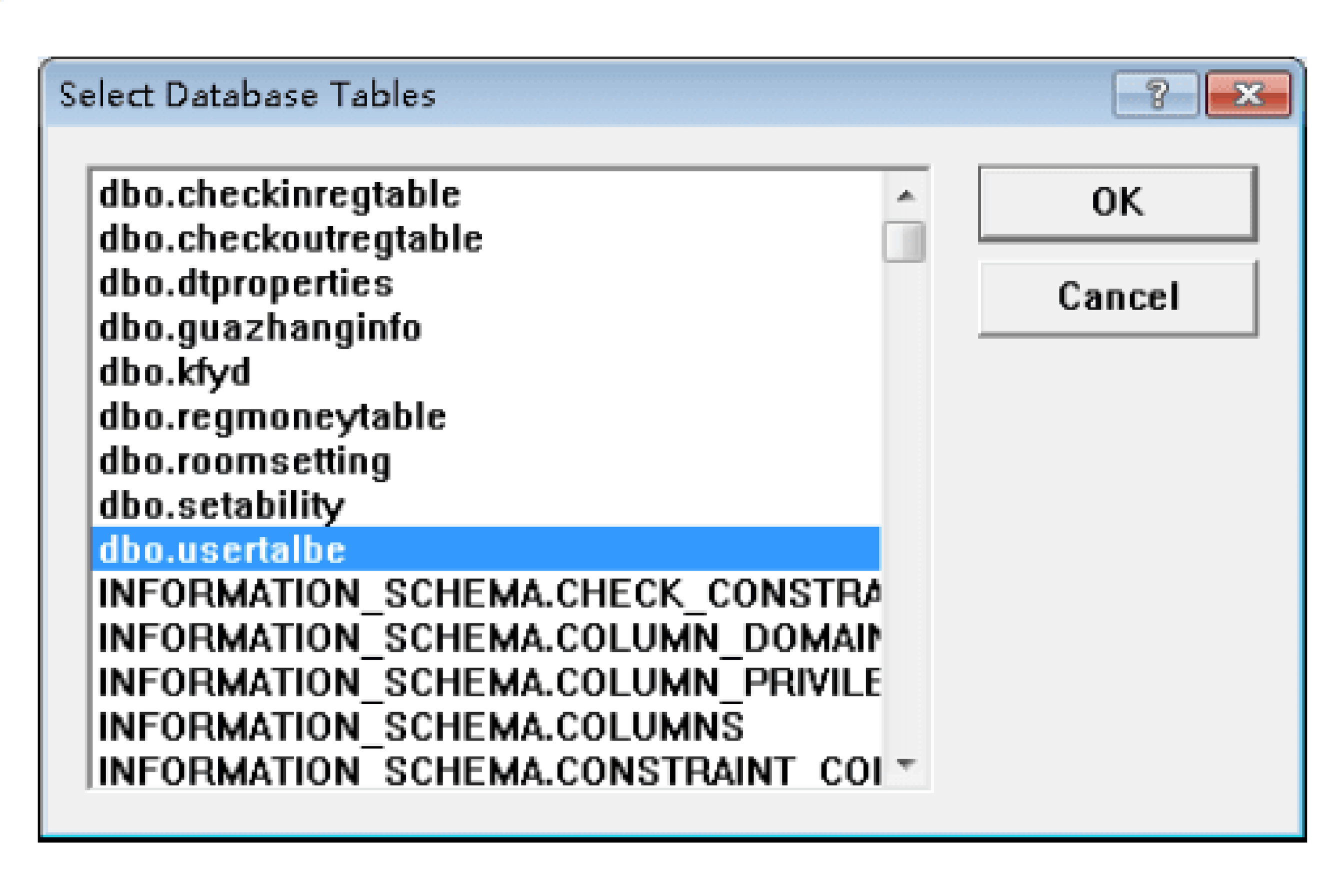


图 24.18　Select Database Tables 对话框

选择所要关联的数据表，因为是操作员登录信息，所以选择 dbo.usertable 数据表，单击 OK 按钮完成映射。

可以看到已经创建了一个新类 CUserset，其头文件的关键代码如下：

```
class CUserset : public CRecordset
{
public:
	CUserset(CDatabase* pDatabase = NULL);
	DECLARE_DYNAMIC(CUserset)
// Field/Param Data
	//{{AFX_FIELD(CUserset, CRecordset)
	CString m_user_name;
	CString m_user_pwd;
	//}}AFX_FIELD
// Overrides
	// ClassWizard generated virtual function overrides
```



```
    //{{AFX_VIRTUAL(CUserset)
    public:
    virtual CString GetDefaultConnect();                    //默认连接字符串
    virtual CString GetDefaultSQL();                        //获取默认的 SQL 支持
    virtual void DoFieldExchange(CFieldExchange* pFX);      //RFX 支持
    //}}AFX_VIRTUAL
// Implementation
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
};
```

（5）单击“确定”按钮可以登录到系统主界面，此按钮的相应函数如下：

```
void CLoginDlg::OnOK()
{
    CString sqlStr;
    UpdateData(true);
    if(m_username.IsEmpty())                             //判断用户名是否为空
    {
        AfxMessageBox("请输入用户名!");
        return;
    }
    //创建查询语句
    sqlStr="SELECT * FROM usertalbe WHERE user_name='";
    sqlStr+=m_username;
    sqlStr+="'";
    sqlStr+="AND user_pwd='";
    sqlStr+=m_password;
    sqlStr+="'";
    //打开数据库
    if(!myuserset.Open(AFX_DB_USE_DEFAULT_TYPE,sqlStr))
    {
        AfxMessageBox("user 表打开失败!");
        return;
    }
    loguserid=m_username;                               //保存操作员 ID，其他窗口中会用到该数据

    if(!myuserset.IsEOF())                              //关闭数据库连接
    {
        myuserset.Close();
        CDialog::OnOK();
    }
    else
    {                                                   //给出错误提示
```

```
            AfxMessageBox("登录失败!");
            m_username=_T("");
            m_password=_T("");
            UpdateData(false);                                  //更新显示
            myuserset.Close();                                  //关闭数据库连接
            return;
        }
}
```

（6）为了按下 Enter 键时控制输入焦点，故加入 PreTranslateMessage 方法，代码如下：

```
BOOL CLoginDlg::PreTranslateMessage(MSG* pMsg)
{
    if(pMsg->message==WM_KEYDOWN&&pMsg->wParam==VK_RETURN)
    {
        DWORD def_id=GetDefID();
         if(def_id!=0)
         {
            //MSG 消息的结构中的 hwnd 存储的是接收该消息的窗口句柄
            CWnd *wnd=FromHandle(pMsg->hwnd);
                char class_name[16];
            if(GetClassName(wnd->GetSafeHwnd(),class_name,sizeof(class_name))!=0)
            {
                DWORD style=::GetWindowLong(pMsg->hwnd,GWL_STYLE);
                if((style&ES_MULTILINE)==0)
                {
                    if(strnicmp(class_name,"edit",5)==0)
                    {   //将焦点设置到默认按钮上
                        GetDlgItem(LOWORD(def_id))->SetFocus();
                        pMsg->wParam=VK_TAB;            //重载 Enter 键消息为 Tab 键消息
                     }
                }
            }
         }
    }
    return CDialog::PreTranslateMessage(pMsg);
}
```

（7）登录模块与数据库连接代码如下：

```
BOOL CLoginDlg::OnInitDialog()
{
     CDialog::OnInitDialog();

     // TODO: Add extra initialization here
     // 使用 ADO 创建数据库记录集
     m_pRecordset.CreateInstance(__uuidof(Recordset));
```

```
    _variant_t var;
    CString struser;
    // 在 ADO 操作中建议语句中要常用 try...catch()来捕获错误信息
    try
    {                                                   //打开数据库
        m_pRecordset->Open("SELECT * FROM usertalbe",                  // 查询表中所有字段
                        theApp.m_pConnection.GetInterfacePtr(), // 获取库接库的 IDispatch 指针
                        adOpenDynamic,
                        adLockOptimistic,
                        adCmdText);
    }
    catch(_com_error *e)                        //捕获打开数据库可能发生的异常情况并实时显示提示
    {
        AfxMessageBox(e->ErrorMessage());
    }
    try
    {
        if(!m_pRecordset->BOF)                  //判断指针是否在数据集最后
            m_pRecordset->MoveFirst();
        else
        {                                       //提示错误，无数据
            AfxMessageBox("表内数据为空");
            return false;
        }
        //读取数据
        while(!m_pRecordset->adoEOF)
        {
            var = m_pRecordset->GetCollect("user_name");
            if(var.vt != VT_NULL)
                struser = (LPCSTR)_bstr_t(var);
            m_usernamectr.AddString(struser);   //从数据库获得的内容给变量赋值

            m_pRecordset->MoveNext();           //移动数据指针
        }
    }
    catch(_com_error *e)                        //捕获异常
    {
        AfxMessageBox(e->ErrorMessage());
    }
    // 关闭记录集
    m_pRecordset->Close();
    m_pRecordset = NULL;
    //更新显示
    UpdateData(false);
    return TRUE;
}
```

（8）在登录界面中，需要对图片有限制，在 LoginDlg.cpp 文件中，写入如下代码：

```
void CLoginDlg::OnPaint()
{
	CPaintDC dc(this);
	CBitmap bit;
	CDC memDC;
	CRect rect;
	this->GetClientRect(&rect);

	bit.LoadBitmap(IDB_LOGINBK);

	BITMAP bmpInfo;
	bit.GetBitmap(&bmpInfo);
	int imgWidth = bmpInfo.bmWidth;
	int imgHeight = bmpInfo.bmHeight;
	memDC.CreateCompatibleDC(&dc);
	memDC.SelectObject(&bit);
	dc.StretchBlt(0,0,rect.Width(),rect.Height(),&memDC,0,0,imgWidth,imgHeight,SRCCOPY);
	memDC.DeleteDC();
	bit.DeleteObject();
}
```

24.6 客房预订模块设计

24.6.1 客房预订模块概述

住宿管理模块包括客房预订、住宿登记、追加押金、调房登记、退宿结账等功能子模块。下面详细介绍客房预订子模块的设计。客房预订模块用于实现客房预订的功能，主要登记客户的姓名、证件、证件号码和预住日期等信息，是为预订客户提供服务的模块。其运行界面如图 24.19 所示。

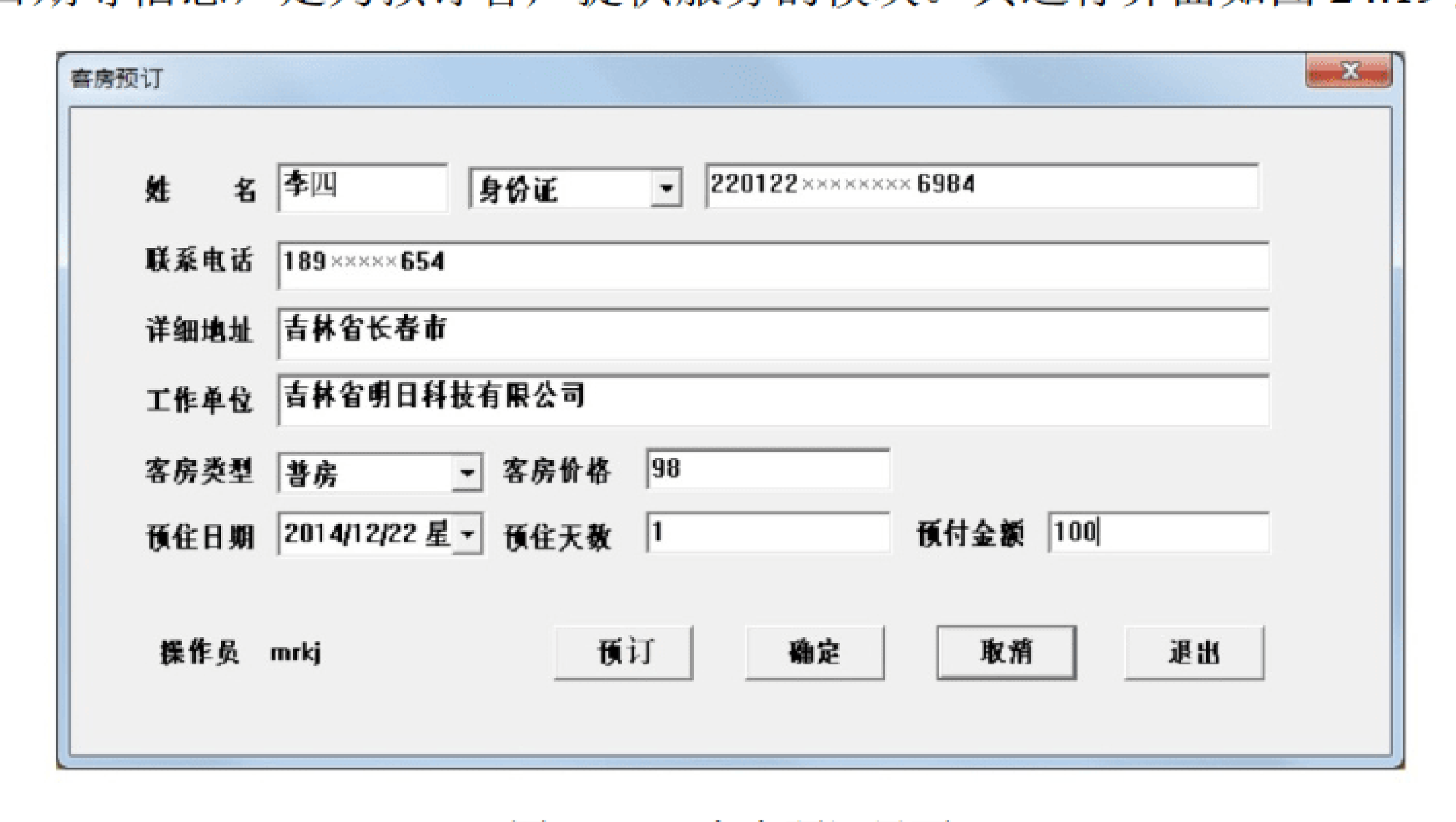


图 24.19　客房预订界面



24.6.2　客房预订模块技术分析

客房预订模块实现将预订客房信息插入到数据表中，主要是通过打开记录集，然后使用 AddNew 方法向数据表中插入一条新记录来实现对客房预订信息的添加。AddNew 方法用于向记录集中添加一个空行，然后设置这个空行的每个字段值，从而能够实现将一条记录添加到数据表中。

24.6.3　客房预订模块实现过程

（1）选择 Insert/Resource 命令，打开插入资源对话框，选择 Dialog 选项，单击新建按钮，插入新的对话框。

（2）利用类向导为此对话框资源设置属性。在 Name 文本框中输入对话框类名，如 CRoomprebookdlg，在 Base class 下拉列表框中选择一个基类，这里为 CDialog，单击确定按钮创建对话框。

（3）在工作区的资源视图中选择新创建的对话框，向对话框中添加静态文本、下拉列表框、编辑框、按钮和日期选择控件等资源。各个主要控件的 ID 和属性如表 24.4 所示。

表 24.4　主要控件的 ID 和属性

控件 ID	变　量	控件 ID	变　量
IDC_COMBOprebookidkind	m_prebookidkind	IDC_prebookidnumber	m_prebookidnumber
IDC_COMBOroomkind	m_prebookroomkind	IDC_prebookname	m_prebookname
IDC_DATETIMEPICKERprecheckindate	m_prebookcheckindate	IDC_prebooktelnumber	m_prebooktelnumber
IDC_prebookaddr	m_prebookaddr	IDC_prebookworkcompany	m_prebookworkcompany
IDC_prebookdays	m_prebookdays	IDC_roommoney	m_prebookroommoney
IDC_prebookhandinmoney	m_prebookhandinmoney	IDC_STATICshowuser	m_showuser

（4）在其对应的头文件 Roomprebookdlg.h 中添加以下声明代码：

```
…
…
CString gustname;
CString gustaddr;

CString zhengjian;
CString zhengjian_number;
CString checkinreg_reason;
_ConnectionPtr m_pConnection;
_CommandPtr m_pCommand;
_RecordsetPtr m_pRecordset;
```

确定预订客房，单击“确定”按钮向数据库中插入预订记录，其响应函数如下：

```
void CRoomprebookdlg::OnOK()
{
    UpdateData(true);
    /*
     *  检查身份证的号码是否为 15 位或者为 18 位
    */
    CString strCertifyCode;                                         //证件号码
    //获得证件号码
    int nCertifyCodeLength=m_prebookidnumber.GetLength();          //获得证件号码的长度
    if(nCertifyCodeLength!=15&&nCertifyCodeLength!=18)
    {
        if(m_prebookidkind=="身份证")
        {                                                           //若选择的是身份证
            MessageBox("你的身份证的号码的位数不正确!\n 应该为 15 位或者 18 位!",
                "身份证错误",MB_OK);
            return ;
        }
    }
    m_pRecordset.CreateInstance(__uuidof(Recordset));
    //在 ADO 操作中建议语句中要常用 try...catch()来捕获错误信息
    try
    {                                                               //打开数据表
        m_pRecordset->Open("SELECT * FROM kfyd",                    //查询表中所有字段
        theApp.m_pConnection.GetInterfacePtr(),                     //获取库接库的 IDispatch 指针
                            adOpenDynamic,
                            adLockOptimistic,
                            adCmdText);
    }
    catch(_com_error *e)                                            //捕获异常情况
    {
        AfxMessageBox(e->ErrorMessage());
    }
       try
       {
        //写入各字段值
        m_pRecordset->AddNew();
        //向数据表“姓名”字段写入数据
        m_pRecordset->PutCollect("姓名",_variant_t(m_prebookname));
        //向数据表“身份证号”字段写入数据
        m_pRecordset->PutCollect("身份证号", _variant_t(m_prebookidnumber));
        //向数据表“联系电话”字段写入数据
        m_pRecordset->PutCollect("联系电话", _variant_t(m_prebooktelnumber));
        //向数据表“详细地址”字段写入数据
        m_pRecordset->PutCollect("详细地址", _variant_t(m_prebookaddr));
```

```
//向数据表“工作单位”字段写入数据
m_pRecordset->PutCollect("工作单位", _variant_t(m_prebookworkcompany));
//向数据表“客房类型”字段写入数据
m_pRecordset->PutCollect("客房类型", _variant_t(m_prebookroomkind));
//向数据表“客房价格”字段写入数据
m_pRecordset->PutCollect("客房价格", _variant_t(m_prebookroommoney));
CString checkindate;
int nYear,nDay,nMonth;
int nhour,nmin,nsecond;
CString sYear,sDay,sMonth;
nYear=m_prebookcheckindate.GetYear();                    //提取年份
nDay=m_prebookcheckindate.GetDay();                      //提取日
nMonth=m_prebookcheckindate.GetMonth();                  //提取月份
sYear.Format("%d",nYear);                                //转换为字符串
sDay.Format("%d",nDay);                                  //转换为字符串
sMonth.Format("%d",nMonth);                              //转换为字符串
//格式化时间
checkindate.Format("%s-%s-%s",sYear,sMonth,sDay);
//向数据表“预住日期”字段写入数据
m_pRecordset->PutCollect("预住日期",_variant_t(checkindate));
//向数据表“预住天数”字段写入数据
m_pRecordset->PutCollect("预住天数", _variant_t(m_prebookdays));
//向数据表中“预付金额”字段写入数据
m_pRecordset->PutCollect("预付金额", _variant_t(m_prebookhandinmoney));
CString nowdate,nowtime;
CTime tTime;
tTime=tTime.GetCurrentTime();
nYear=tTime.GetYear();                                   //提取年份
nDay=tTime.GetDay();                                     //提取日
nMonth=tTime.GetMonth();                                 //提取月份
sYear.Format("%d",nYear);                                //转换为字符串
sDay.Format("%d",nDay);                                  //转换为字符串
sMonth.Format("%d",nMonth);                              //转换为字符串
//格式化时间
nowdate.Format("%s-%s-%s",sYear,sMonth,sDay);
CString shour,smin,ssecond;
nhour=tTime.GetHour();                                   //提取小时
nmin=tTime.GetMinute();                                  //提取分钟
nsecond=tTime.GetSecond();                               //提取秒
shour.Format("%d",nhour);                                //转换为字符串
smin.Format("%d",nmin);                                  //转换为字符串
ssecond.Format("%d",nsecond);                            //转换为字符串
//格式化时间
nowtime.Format("%s:%s:%s",shour,smin,ssecond);
m_pRecordset->PutCollect("日期", _variant_t(nowdate));
m_pRecordset->PutCollect("时间", _variant_t(nowtime));
```

```
        //向数据表“证件名称”字段写入数据
        m_pRecordset->PutCollect("证件名称", _variant_t(m_prebookidkind));
        //更新数据表
        m_pRecordset->Update();
        AfxMessageBox("预订成功!");
       }
        catch(_com_error *e)                                    //抛出异常情况，并显示
        {
        AfxMessageBox(e->ErrorMessage());
        }
        //关闭记录集
        m_pRecordset->Close();
        m_pRecordset = NULL;
}
```

在预订房间时，需要选择客房类型，实现的具体代码如下：

```
void CRoomprebookdlg::OnCloseupCOMBOroomkind()
{
    // TODO: Add your control notification handler code here
   //获得输入值
    UpdateData(true);
    roomkind=m_prebookroomkind;
    //如果客房类型是标房
    if(m_prebookroomkind=="标房")
    {
        m_prebookroommoney="138";
    }
    //如果客房类型是普房
    if(m_prebookroomkind=="普房")
    {
        m_prebookroommoney="98";

    }
    //如果客房类型是双人间
    if(m_prebookroomkind=="双人间")
    {
        m_prebookroommoney="168";

    }
    //如果客房类型是套房
    if(m_prebookroomkind=="套房")
    {
        m_prebookroommoney="268";

    }
    //更新显示
```

```
        UpdateData(false);
}
```

如果客户想在入住之前换房间，就需要重新修改预订信息，实现此功能的代码如下：

```
void CRoomprebookdlg::Oncancelprebookroom()
{
        // TODO: Add your control notification handler code here
        //输入变量初始化
        m_prebookidkind = _T("");
        m_prebookroomkind = _T("");
        m_prebookcheckindate = 0;
        m_prebookaddr = _T("");
        m_prebookdays = _T("");
        m_prebookhandinmoney = _T("");
        m_prebookidnumber = _T("");
        m_prebookname = _T("");
        m_prebooktelnumber = _T("");
        m_prebookworkcompany = _T("");
        m_prebookroommoney = _T("");
        CTime tTime;
        tTime=tTime.GetCurrentTime();
        //设置登记的默认时间
       m_prebookcheckindate=tTime;
        //更新显示
        UpdateData(false);

}

BOOL CRoomprebookdlg::OnInitDialog()
{
        CDialog::OnInitDialog();

        // TODO: Add extra initialization here
        m_showuser=loguserid;
        enable(0);                                    //更新输入框状态

        CTime tTime;
        tTime=tTime.GetCurrentTime();
        //设置登记的默认时间
       m_prebookcheckindate=tTime;
        UpdateData(false);

        return TRUE;
}

void CRoomprebookdlg::OnBtnroomyuding()
```

```
{
	// TODO: Add your control notification handler code here
	enable(1);		//更新输入框状态

}
void CRoomprebookdlg::enable(bool bEnabled)
{
	//更改输入框等控件状态，方便使用，防止错误操作
	GetDlgItem(IDC_COMBOprebookidkind)->EnableWindow(bEnabled);
	GetDlgItem(IDC_COMBOroomkind)->EnableWindow(bEnabled);
	GetDlgItem(IDC_DATETIMEPICKERprecheckindate)->EnableWindow(bEnabled);
	GetDlgItem(IDC_prebookaddr)->EnableWindow(bEnabled);
	GetDlgItem(IDC_prebookdays)->EnableWindow(bEnabled);
	GetDlgItem(IDC_prebookhandinmoney)->EnableWindow(bEnabled);
	GetDlgItem(IDC_prebookidnumber)->EnableWindow(bEnabled);
	GetDlgItem(IDC_prebookname)->EnableWindow(bEnabled);
	GetDlgItem(IDC_prebooktelnumber)->EnableWindow(bEnabled);
	GetDlgItem(IDC_prebookworkcompany)->EnableWindow(bEnabled);
	GetDlgItem(IDC_roommoney)->EnableWindow(bEnabled);
	GetDlgItem(IDOK)->EnableWindow(bEnabled);
	GetDlgItem(IDcancelprebookroom)->EnableWindow(bEnabled);

}
```

24.7　追加押金模块设计

24.7.1　追加押金模块概述

追加押金模块是为方便客户追加预交的住房押金而设计的，在此子对话框中只要选择客户的凭证号码，然后输入追加的金额就可以轻松地完成追加操作，其运行界面如图 24.20 所示。

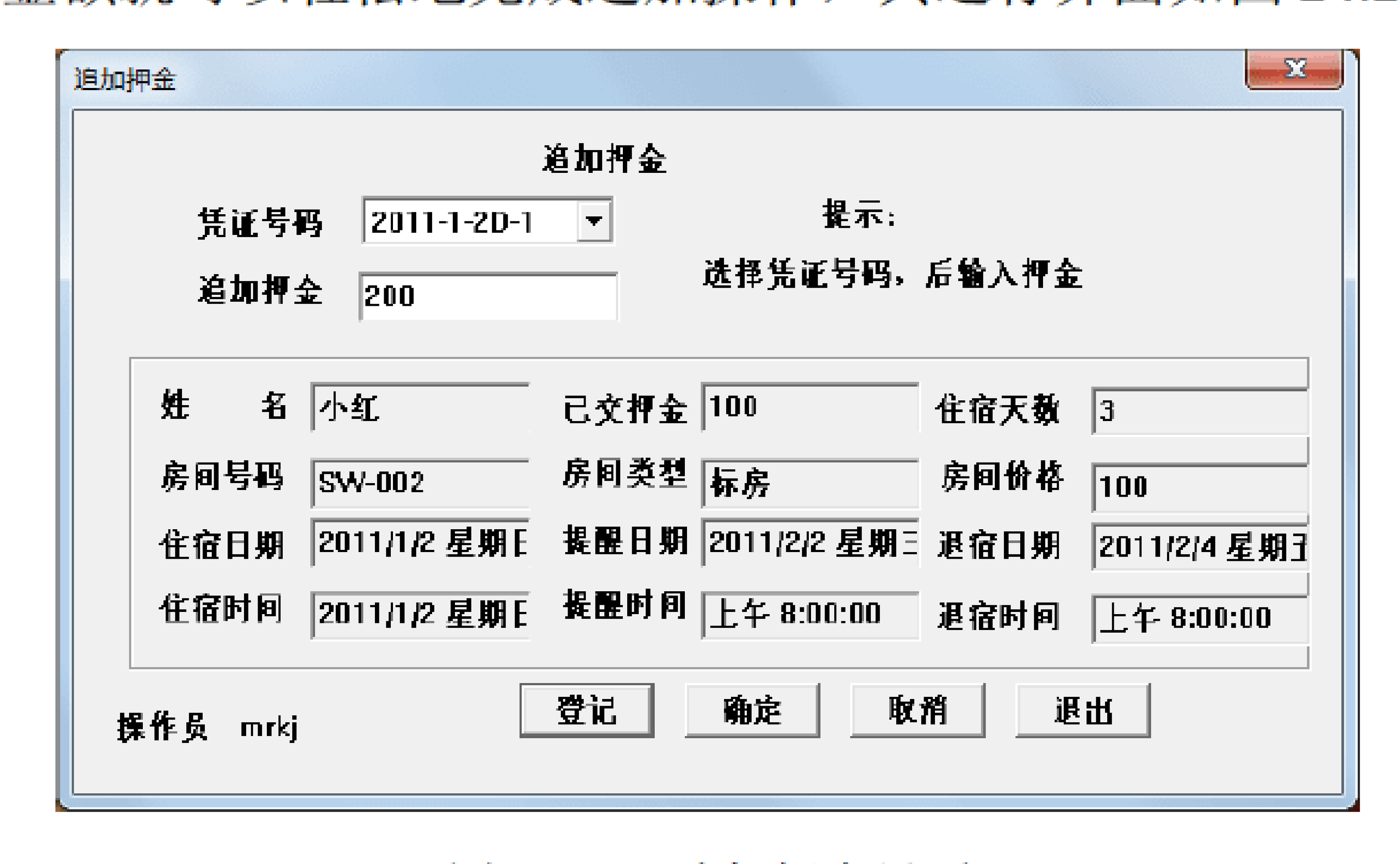


图 24.20　追加押金界面



24.7.2　追加押金模块技术分析

追加押金模块用于将追加押金信息记录到数据表中。在打开窗体时，凭证号码组合框中自动显示了当前数据库中的凭证号码，可直接在此选择一个凭证号码。这个凭证号码是在窗体初始化时添加到组合框中的。通过查询符合条件的记录，使用循环语句将记录添加到组合框中，其实现代码如下：

```
while(!m_pRecordset->adoEOF)
{
        var = m_pRecordset->GetCollect("凭证号码");
        if(var.vt != VT_NULL)
                strregnumber = (LPCSTR)_bstr_t(var);
        m_addmoney_regnumberctr.AddString(strregnumber);
        m_pRecordset->MoveNext();                                   //移动记录指针到下一条记录
}
```

24.7.3　追加押金模块实现过程

（1）选择 Insert/Resource 命令，打开插入资源对话框，选择 Dialog 选项，单击新建按钮，插入新的对话框。

（2）利用类向导为此对话框资源设置属性。在 Name 文本框中输入对话框类名，如 CAddmoneydlg，在 Base class 下拉列表框中选择一个基类，这里为 CDialog，单击确定按钮创建对话框。

（3）在工作区的资源视图中选择新创建的对话框，向对话框中添加静态文本、下拉列表框、编辑框、按钮和时间日期选择控件等资源。各个控件的 ID 和属性如表 24.5 所示。

表 24.5　各控件的 ID 和属性

控件 ID	对 应 变 量	控件 ID	对 应 变 量
IDC_COMBO_regnumber	m_addmoney_regnumberctr	IDC_EDIT_roomnumber	m_addmoney_roomnumber
IDC_COMBO_regnumber	m_addmoney_regnumber	IDC_EDIT_alarmdate	m_addmoney_alarmdate
IDC_EDIT_name	m_addmoney_name	IDC_EDIT_alarmtime	m_addmoney_alarmtime
IDC_EDIT_outdate	m_addmoney_outdate	IDC_EDIT_checkdays	m_addmoney_checkdays
IDC_EDIT_outtime	m_addmoney_outtime	IDC_EDIT_indate	m_addmoney_indate
IDC_EDIT_prehandmoney	m_addmoney_prehandmoney	IDC_EDIT_intime	m_addmoney_intime
IDC_EDIT_roomlevel	m_addmoney_roomlevel	IDC_addmoney	m_addmoney
IDC_EDIT_roommoney	m_addmoney_roommoney	IDC_STATICshowuser	m_showuser

（4）在对应类的头文件 Addmoneydlg.h 中声明以下变量：

```
_ConnectionPtr m_pConnection;
_CommandPtr m_pCommand;
_RecordsetPtr m_pRecordset;
```

```
_RecordsetPtr m_pRecordsetout;
```

对话框的初始化函数完成住宿客户凭证号码的准备等其他的初始化工作，该对话框类的初始化函数如下：

```
BOOL CAddmoneydlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    //使用 ADO 创建数据库记录集
    m_pRecordset.CreateInstance(__uuidof(Recordset));
    _variant_t var;
    CString strregnumber;
    //在 ADO 操作中建议语句中要常用 try...catch()来捕获错误信息
    try
    {
        m_pRecordset->Open("SELECT * FROM checkinregtable",        //查询表中所有字段
        theApp.m_pConnection.GetInterfacePtr(),                     //获取库接库的 IDispatch 指针
        adOpenDynamic,
        adLockOptimistic,
        adCmdText);
    }
    catch(_com_error *e)                                            //抛出异常
    {
        AfxMessageBox(e->ErrorMessage());
    }
    try
    {
        if(!m_pRecordset->BOF)                                      //判断指针是否在数据集最后
            m_pRecordset->MoveFirst();
        else
        {
            AfxMessageBox("表内数据为空");
            return false;
        }
        while(!m_pRecordset->adoEOF)
        {
            var = m_pRecordset->GetCollect("凭证号码");
            if(var.vt != VT_NULL)
                strregnumber = (LPCSTR)_bstr_t(var);
            m_addmoney_regnumberctr.AddString(strregnumber);
            m_pRecordset->MoveNext();                               //移动记录指针到下一条记录
        }
    }
    catch(_com_error *e)                                            //如果读数异常，给出提示
    {
        AfxMessageBox(e->ErrorMessage());
```

```
	}
	//关闭记录集
	m_pRecordset->Close();
	m_pRecordset = NULL;
	m_showuser=loguserid;
	//更新显示
	UpdateData(false);
	enable(0);
	return TRUE;
}
```

完成追加押金操作的“确定”按钮的处理函数，代码如下：

```
void CAddmoneydlg::OnOK()
{
	UpdateData(true);
	//获得输入框内的输入数据
	m_pRecordsetout.CreateInstance(__uuidof(Recordset));
	CString strsqlstore;
	strsqlstore.Format("SELECT * FROM checkinregtable where 凭证号码='%s'",m_addmoney_regnumber);

	try												//连接数据库
	{
		m_pRecordsetout->Open(_variant_t(strsqlstore),			//查询表中所有字段
		theApp.m_pConnection.GetInterfacePtr(),				//获取数据库的 IDispatch 指针
						adOpenDynamic,
						adLockOptimistic,
						adCmdText);
	}
	catch(_com_error *e)									//捕获连接数据库异常
	{
		AfxMessageBox(e->ErrorMessage());
	}
	try												//更新数据库
	{
		float theaddedmoney=atof(m_addmoney_prehandmoney)+atof(m_addmoney);
		char strtheaddedmoney[50];
		_gcvt(theaddedmoney, 4, strtheaddedmoney );				//格式转换
		//写入数据表
		m_pRecordsetout->PutCollect("预收金额", _variant_t(strtheaddedmoney));
		m_pRecordsetout->Update();
		//更新数据库完毕
		AfxMessageBox("追加成功!");
	}
	catch(_com_error *e)									//捕获连接数据库异常
	{
```

```
            AfxMessageBox(e->ErrorMessage());
      }
}
```

在追加押金时，需要登记一下，“登记”按钮处理的函数代码如下：

```
void CAddmoneydlg::OnCloseupCOMBOregnumber()
{
      // TODO: Add your control notification handler code here
      _variant_t var;
      // 使用 ADO 创建数据库记录集
      m_pRecordset.CreateInstance(__uuidof(Recordset));
      // 在 ADO 操作中建议语句中要常用 try...catch()来捕获错误信息

      UpdateData(true);

     m_addmoney_regnumberctr.GetWindowText(m_addmoney_regnumber);

      CString strsql;
     strsql.Format("SELECT * FROM checkinregtable where 凭证号码='%s'",m_addmoney_regnumber);
      try
      {                                                               // 打开数据表
            m_pRecordset->Open(_variant_t(strsql),                    // 查询表中所有字段
                                theApp.m_pConnection.GetInterfacePtr(), // 获取库接库的 IDispatch 指针
                                adOpenDynamic,
                                adLockOptimistic,
                                adCmdText);
      }
      catch(_com_error *e)                                            // 捕获异常
      {
            AfxMessageBox(e->ErrorMessage());
      }


      try
      {                                                               //判断指针是否在数据集最后
            if(!m_pRecordset->BOF)
                  m_pRecordset->MoveFirst();
            else
            {
                  AfxMessageBox("表内数据为空");
                  return ;
            }

            // 读取姓名
            var = m_pRecordset->GetCollect("姓名");
            if(var.vt != VT_NULL)
                  m_addmoney_name = (LPCSTR)_bstr_t(var);
            // 读取房间号
            var = m_pRecordset->GetCollect("房间号");
```

```
    if(var.vt != VT_NULL)
        m_addmoney_roomnumber = (LPCSTR)_bstr_t(var);
    // 读取客房类型
    var = m_pRecordset->GetCollect("客房类型");
    if(var.vt != VT_NULL)
        m_addmoney_roomlevel = (LPCSTR)_bstr_t(var);
    // 读取客房价格
    var = m_pRecordset->GetCollect("客房价格");
    if(var.vt != VT_NULL)
        m_addmoney_roommoney = (LPCSTR)_bstr_t(var);
    // 读取住宿天数
    var = m_pRecordset->GetCollect("住宿天数");
    if(var.vt != VT_NULL)
        m_addmoney_checkdays = atof((LPCSTR)_bstr_t(var));
    CString checkindate;// 读取住宿日期
    var = m_pRecordset->GetCollect("住宿日期");
    if(var.vt != VT_NULL)
        m_addmoney_indate = (LPCSTR)_bstr_t(var);
   // 读取住宿时间
    var = m_pRecordset->GetCollect("住宿时间");
    if(var.vt != VT_NULL)
        m_addmoney_intime = (LPCSTR)_bstr_t(var);

    // 读取预收金额
    var = m_pRecordset->GetCollect("预收金额");
    if(var.vt != VT_NULL)
        m_addmoney_prehandmoney = (LPCSTR)_bstr_t(var);
    else
      m_addmoney_prehandmoney="000";
    // 读取退宿日期

    var = m_pRecordset->GetCollect("退宿日期");
    if(var.vt != VT_NULL)
        m_addmoney_outdate = (LPCSTR)_bstr_t(var);
    // 读取退宿时间
    var = m_pRecordset->GetCollect("退宿时间");
    if(var.vt != VT_NULL)
        m_addmoney_outtime = (LPCSTR)_bstr_t(var);
    // 读取提醒日期
    var = m_pRecordset->GetCollect("提醒日期");
    if(var.vt != VT_NULL)
        m_addmoney_alarmdate = (LPCSTR)_bstr_t(var);
    // 读取提醒时间
    var = m_pRecordset->GetCollect("提醒时间");
    if(var.vt != VT_NULL)
            m_addmoney_alarmtime   = (LPCSTR)_bstr_t(var);
    // 更新显示
    UpdateData(false);
```

```
            // 从数据库内读取数据完毕
        }
        catch(_com_error *e)
        {       //如果读数异常，给出提示
                AfxMessageBox(e->ErrorMessage());
        }
        // 关闭记录集
        m_pRecordset->Close();
        m_pRecordset = NULL;
        //更新显示
        UpdateData(false);
}
```

在追加押金时，需要将对应的输入框初始化设置，具体代码如下：

```
void CAddmoneydlg::Onaddmoney()
{
        // TODO: Add your control notification handler code here
        //初始化输入框内容
        m_addmoney_regnumber = _T("");
        m_addmoney_name = _T("");
        m_addmoney_outdate = _T("");
        m_addmoney_outtime = _T("");
        m_addmoney_prehandmoney = _T("");
        m_addmoney_roomlevel = _T("");
        m_addmoney_roommoney = _T("");
        m_addmoney_roomnumber = _T("");
        m_addmoney_alarmdate = _T("");
        m_addmoney_alarmtime = _T("");
        m_addmoney_checkdays = 0.0f;
        m_addmoney_indate = _T("");
        m_addmoney_intime = _T("");
        m_addmoney = _T("");
        UpdateData(false);
}
```

其他函数的处理代码见源程序。

24.8　调房登记模块设计

24.8.1　调房登记模块概述

调房登记模块是为实现客户调房而设计的，有的客户可能在住宿期间要求调换房间，该模块可以通过选择原房间号和目标房间号实现调房操作，其运行界面如图 24.21 所示。

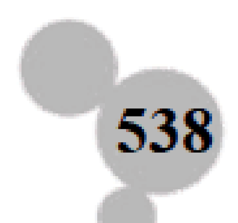




图 24.21　调房登记界面

24.8.2　调房登记模块技术分析

调房登记模块根据所选择的房间号，在住宿登记表中查询相关记录。如果查询到记录，则将记录显示在窗体上，然后输入目标房间号记录，将记录保存到住宿登记表中。根据房间号读取相关的住宿信息的主要代码如下：

```
if(!m_pRecordset->BOF)                                  //判断指针是否在数据集最后
	m_pRecordset->MoveFirst();
else
{
	AfxMessageBox("表内数据为空");
	return ;
}

//从数据表中读取客房价格字段
var = m_pRecordset->GetCollect("客房价格");
if(var.vt != VT_NULL)
	m_changeroom_roommoney = (LPCSTR)_bstr_t(var);
```

24.8.3　调房登记模块实现过程

（1）选择 Insert/Resource 命令，打开插入资源对话框，选择 Dialog 选项，单击新建按钮，插入新的对话框。

（2）利用类向导为此对话框资源设置属性。在 Name 文本框中输入对话框类名，如 CChangeroomdlg，在 Base class 下拉列表框中选择一个基类，这里为 CDialog，单击确定按钮创建对话框。

（3）在工作区的资源视图中选择新创建的对话框，向对话框中添加静态文本、下拉列表框、编辑框和按钮等资源。各个控件的 ID 和属性如表 24.6 所示。

表 24.6　各控件的 ID 和属性

控件 ID	对 应 变 量	控件 ID	对 应 变 量
IDC_COMBO_destroom	m_destroomctr	IDC_changeroom_idnumber	m_changeroom_idnumber
IDC_COMBO_sourceroom	m_sourceroomctr	IDC_changeroom_name	m_changeroom_name
IDC_COMBO_sourceroom	m_sourceroom	IDC_changeroom_roommoney	m_changeroom_roommoney
IDC_COMBO_destroom	m_destroom	IDC_changeroomdlg_regnumber	m_changeroom_regnumber
IDC_changeroom_beizhu	m_changeroom_beizhu	IDC_STATICshowuser	m_showuser
IDC_changeroom_idkind	m_changeroom_idkind		

（4）在对应类的头文件 Changeroomdlg.h 中声明以下变量：

```
void enable(bool bEnabled);
CString destroomlevel;
_ConnectionPtr m_pConnection;
_CommandPtr m_pCommand;
_RecordsetPtr m_pRecordset;
_RecordsetPtr m_pRecordsetout;
```

该对话框类的初始化函数如下：

```
BOOL CChangeroomdlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    //使用 ADO 创建数据库记录集
    m_pRecordset.CreateInstance(__uuidof(Recordset));
    _variant_t var;
    CString strroomnumber;
    //在 ADO 操作中建议语句中要常用 try...catch()来捕获错误信息
    try
    {
        m_pRecordset->Open("SELECT * FROM checkinregtable",      //查询表中所有字段
        theApp.m_pConnection.GetInterfacePtr(),                  //获取库接库的 IDispatch 指针
                        adOpenDynamic,
                        adLockOptimistic,
                        adCmdText);
    }
    catch(_com_error *e)                                         //捕获连接数据库异常
    {
        AfxMessageBox(e->ErrorMessage());
    }
    try
    {
        if(!m_pRecordset->BOF)                                   //判断指针是否在数据集最后
            m_pRecordset->MoveFirst();
        else
```

```
        {
            AfxMessageBox("表内数据为空");
            return false;
        }
        //从数据库表中读取数据
        while(!m_pRecordset->adoEOF)
        {                                                   //读取房间号
            var = m_pRecordset->GetCollect("房间号");
            if(var.vt != VT_NULL)
            strroomnumber = (LPCSTR)_bstr_t(var);
            m_sourceroomctr.AddString(strroomnumber);       //添加到列表
            m_destroomctr.AddString(strroomnumber);
            m_pRecordset->MoveNext();                       //移动记录集指针
        }
    }
    catch(_com_error *e)                                    //捕获异常
    {
        AfxMessageBox(e->ErrorMessage());
    }
    //关闭记录集
    m_pRecordset->Close();
    m_pRecordset = NULL;
    //获得操作员 ID
    m_showuser=loguserid;
    //显示更新
    UpdateData(false);
    enable(0);
    return TRUE;
}
```

完成调房登记模块的“确定”按钮的处理函数，代码如下：

```
void CChangeroomdlg::OnOK()
{
    UpdateData(true);
    m_pRecordsetout.CreateInstance(__uuidof(Recordset));
    CString strsqlstore;
    strsqlstore.Format("SELECT * FROM checkinregtable where 凭证号码='%s'",m_changeroom_regnumber);
    //打开数据库
    try
    {
        m_pRecordsetout->Open(_variant_t(strsqlstore),      //查询表中所有字段
          //获取库接库的 IDispatch 指针
        theApp.m_pConnection.GetInterfacePtr(),
                                adOpenDynamic,
                                adLockOptimistic,
```

```
                                        adCmdText);
    }
    catch(_com_error *e)
    {//捕获打开数据库时候的异常情况，并给出提示
        AfxMessageBox(e->ErrorMessage());
    }
    try
    {
        //往数据库内写入数据
        CString zhaiyao;
        zhaiyao.Format("从原房间%s 调换到目标房间%s",m_sourceroom,m_destroom);
        m_pRecordsetout->PutCollect("房间号", _variant_t(m_destroom));
        //写入数据表“房间号”字段
        m_pRecordsetout->PutCollect("摘要", _variant_t( zhaiyao));
        //写入数据表“摘要”字段
        m_pRecordsetout->PutCollect("客房价格", _variant_t(m_changeroom_roommoney));
        //写入数据表“客房价格”字段
        m_pRecordsetout->PutCollect("客房类型", _variant_t(destroomlevel));
        //写入数据表“客房类型”字段
        m_pRecordsetout->Update();
        //写入数据完毕，给出提示
        AfxMessageBox("调换成功!");
        //UpdateData(false);
    }
    catch(_com_error *e)//捕获写入数据时候的异常情况，实时显示
    {
        AfxMessageBox(e->ErrorMessage());
    }
}
```

客户要求调房，就需要提供证件等有效信息进行查询确认，实现的具体代码如下：

```
void CChangeroomdlg::OnCloseupCOMBOsourceroom()
{
    // TODO: Add your control notification handler code here
    _variant_t var;
    // 使用 ADO 创建数据库记录集
    m_pRecordset.CreateInstance(__uuidof(Recordset));

    // 在 ADO 操作中建议语句中要常用 try...catch()来捕获错误信息
    UpdateData(true);

    CString strsql;
    strsql.Format("SELECT * FROM checkinregtable where 房间号='%s'",m_sourceroom);
    try
    {                                                              //打开数据库
```

```
        m_pRecordset->Open(_variant_t(strsql),                    //查询表中所有字段
            theApp.m_pConnection.GetInterfacePtr(),               //获取库接库的 IDispatch 指针
            adOpenDynamic,
            adLockOptimistic,
            adCmdText);
    }
    catch(_com_error *e)
    {                                                             //捕获异常
        AfxMessageBox(e->ErrorMessage());
    }
    try
    {
        if(!m_pRecordset->BOF)                                    //判断指针是否在数据集最后
            m_pRecordset->MoveFirst();
        else
        {
            AfxMessageBox("表内数据为空");
            return ;
        }

        // read data from the database table
        //从数据表中读取姓名字段
        var = m_pRecordset->GetCollect("姓名");
        if(var.vt != VT_NULL)
            m_changeroom_name= (LPCSTR)_bstr_t(var);
        //从数据表中读取凭证号码字段
        var = m_pRecordset->GetCollect("凭证号码");
        if(var.vt != VT_NULL)
            m_changeroom_regnumber= (LPCSTR)_bstr_t(var);
        //从数据表中读取证件名称字段
        var = m_pRecordset->GetCollect("证件名称");
        if(var.vt != VT_NULL)
            m_changeroom_idkind   = (LPCSTR)_bstr_t(var);
        //从数据表中读取证件号码字段
        var = m_pRecordset->GetCollect("证件号码");
        if(var.vt != VT_NULL)
            m_changeroom_idnumber = (LPCSTR)_bstr_t(var);
        //从数据表中读取备注字段
        var = m_pRecordset->GetCollect("备注");
        if(var.vt != VT_NULL)
            m_changeroom_beizhu = (LPCSTR)_bstr_t(var);

        UpdateData(false);                                            //更新显示
    }
    catch(_com_error *e)                                              //捕获异常
    {
```

```
            AfxMessageBox(e->ErrorMessage());
        }

        // 关闭记录集
        m_pRecordset->Close();
        m_pRecordset = NULL;
        // 更新显示
        UpdateData(false);
}
```

调房登记选择房间类型等相关信息，实现的具体代码如下：

```
void CChangeroomdlg::OnCloseupCOMBOdestroom()
{
        // TODO: Add your control notification handler code here
        _variant_t var;
        // 使用 ADO 创建数据库记录集
        m_pRecordset.CreateInstance(__uuidof(Recordset));

        // 在 ADO 操作中建议语句中要常用 try...catch()来捕获错误信息

        UpdateData(true);


        CString strsql;
       strsql.Format("SELECT * FROM checkinregtable where 房间号='%s'",m_destroom);
        try                                                             //打开数据库
        {
            m_pRecordset->Open(_variant_t(strsql),                      //查询表中所有字段
                theApp.m_pConnection.GetInterfacePtr(),                 //获取库接库的 IDispatch 指针
                adOpenDynamic,
                adLockOptimistic,
                adCmdText);
        }
        catch(_com_error *e)
        {                                                               //捕获打开数据库时候可能发生的异常情况
            AfxMessageBox(e->ErrorMessage());
        }
        try
        {
            if(!m_pRecordset->BOF)                                      //判断指针是否在数据集最后
                m_pRecordset->MoveFirst();
            else
            {
                AfxMessageBox("表内数据为空");
                return ;
```

```
            }
            // read data from the database table
            //从数据表中读取客房价格字段
            var = m_pRecordset->GetCollect("客房价格");
            if(var.vt != VT_NULL)
                m_changeroom_roommoney = (LPCSTR)_bstr_t(var);
            //从数据表中读取客房类型字段
            var = m_pRecordset->GetCollect("客房类型");
            if(var.vt != VT_NULL)
                destroomlevel = (LPCSTR)_bstr_t(var);
            //读取数据完毕，然后更新显示
            UpdateData(false);
            //更新显示完毕
        }
        catch(_com_error *e)                                        // 捕获异常
        {
            AfxMessageBox(e->ErrorMessage());
        }
        // 关闭记录集
        m_pRecordset->Close();
        m_pRecordset = NULL;
        UpdateData(false);                                          //更新显示
}
```

24.9　小　结

本章的主要内容是根据宾馆、酒店客房管理的实际情况设计一个客房管理系统。通过本章的学习，可以了解一个客房管理系统的开发流程。本章通过详细的讲解及简洁的代码使读者能够更快、更好地掌握数据库管理系统的开发技术，增加读者的实际开发能力和项目经验。

第 25 章　在线考试系统（ASP.NET +SQL Server 2014 实现）

传统考试要求教师打印试卷，安排考试，监考，收集试卷，评改试卷，讲评试卷，以及分析试卷。这是一个漫长而复杂的过程，已经越来越不能满足现代教学的需求。在线考试系统是传统考试的延伸，它可以利用网络的广阔空间，随时随地对学生进行考试，加上数据库技术的利用，大大简化了传统考试的过程。因此在线考试系统是电子化教学不可缺少的一个重要环节。通过本章的学习，可以掌握以下要点：

☑ 验证不同身份的登录用户
☑ 随机抽取试题
☑ 实现系统自动评分
☑ 合理创建后台管理

25.1　开 发 背 景

近年来，计算机技术、网络技术的迅猛发展，给传统办学提出了新的模式。目前，大学和学院都已接入互联网并建成校园网，各校的硬件设施已经比较完善。通过设计和建设网络拓扑架构、网络安全系统、数据库基础结构、信息共享与管理、信息的发布与管理，从而方便管理者、教师和学生间信息发布、信息交流和信息共享。以现代计算机技术、网络技术为基础的数字化教学主要是朝着信息化、网络化、现代化的目标迈进。开发的无纸化在线考试系统，目的在于探索一种以互联网为基础的考试模式。通过这种新的模式，提高了考试工作效率和标准化水平，使学校管理者、教师和学生可以在任何时候、任何地点通过网络进行在线考试。

25.2　系 统 分 析

25.2.1　需求分析

在我国，虽然远程教育已经蓬勃发展起来，但是目前学校与社会上的各种考试大都采用传统的考试方式。在此方式下，组织一次考试至少要经过 5 个步骤，即人工出题、考生考试、人工阅卷、成绩

评估和试卷分析。

显然，随着考试类型的不断增加以及考试要求的不断提高，教师的工作量将会越来越大，并且其工作将是一件十分烦琐和非常容易出错的事情，可以说传统的考试方式已经不能适应现代考试的需要。随着计算机应用的迅猛发展，网络应用不断扩大，人们迫切要求利用这些技术来进行在线考试，以减轻教师的工作负担并提高工作效率，与此同时也提高了考试的质量，从而使考试更趋于公正、客观，更加激发学生的学习兴趣。

25.2.2　系统功能分析

为了保障整个系统的安全性，在线考试系统实现了分类验证的登录模块，通过此模块，可以对不同身份的登录用户进行验证，确保了不同身份的用户操作系统。在抽取试题上，系统使用随机抽取试题的方式，体现了考试的客观与公正。当考生答题完毕之后，提交试卷即可得知本次考试的得分，体现系统的高效性。在后台管理上，分后台管理员管理模块和教师管理模块。其分别适应不同的用户，前者只有系统的高级管理员才能进入，对整个系统进行管理；后者只允许教师登录，教师可以对自己任教的科目试题进行修改，并且可以查看所有参加过自己任教科目的学生成绩。

25.3　系 统 设 计

25.3.1　系统目标

本系统属于小型的在线考试系统，可以从数据库中随机抽取试题，并且可以自动对考生的答案评分。本系统主要实现以下目标：

- ☑ 系统采用人机交互的方式，界面美观友好，信息查询灵活、方便，数据存储安全可靠。
- ☑ 实现从数据库中随机抽取试题。
- ☑ 对用户输入的数据，进行严格的数据检验，尽可能地避免人为错误。
- ☑ 实现对考试结果自动评分。
- ☑ 实现教师和后台管理员对试题信息单独管理。
- ☑ 系统最大限度地实现易维护性和易操作性。

25.3.2　系统功能结构

在线考试系统前台的功能结构如图 25.1 所示。在线考试系统后台的功能结构如图 25.2 所示。

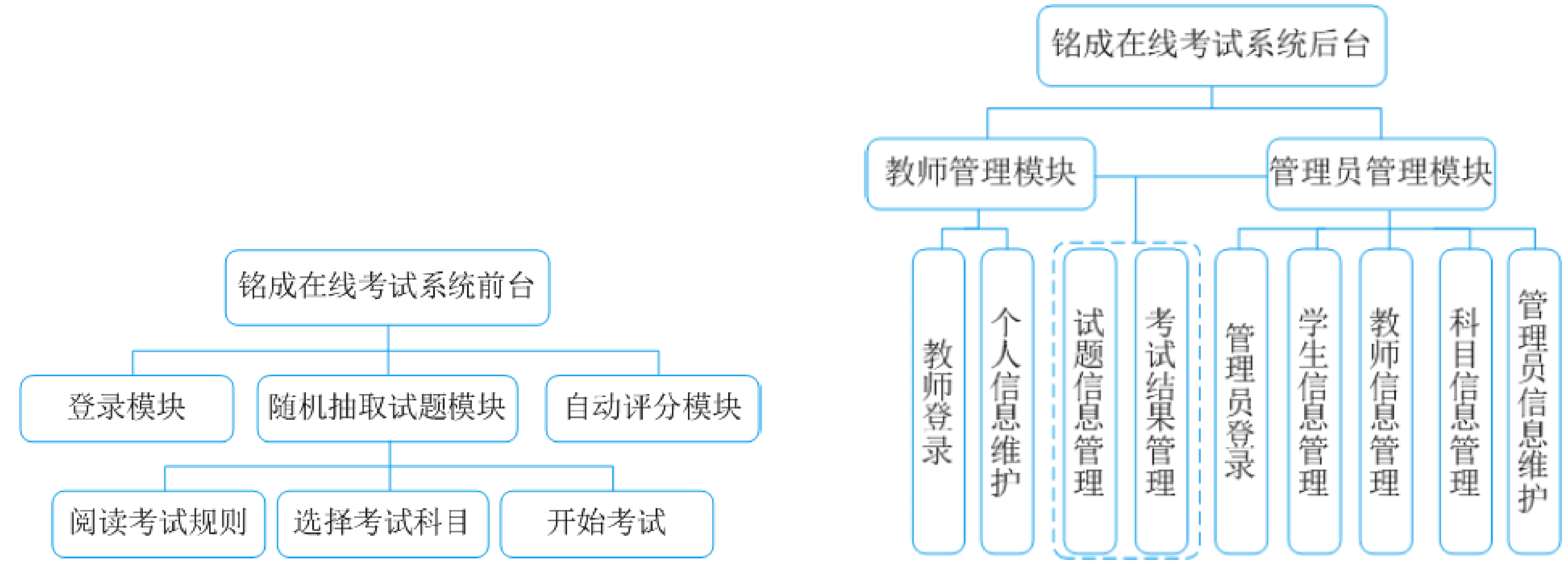


图 25.1　铭成在线考试系统前台的功能结构　　图 25.2　铭成在线考试系统后台的功能结构

25.3.3　业务流程图

在线考试系统的业务流程图如图 25.3 所示。

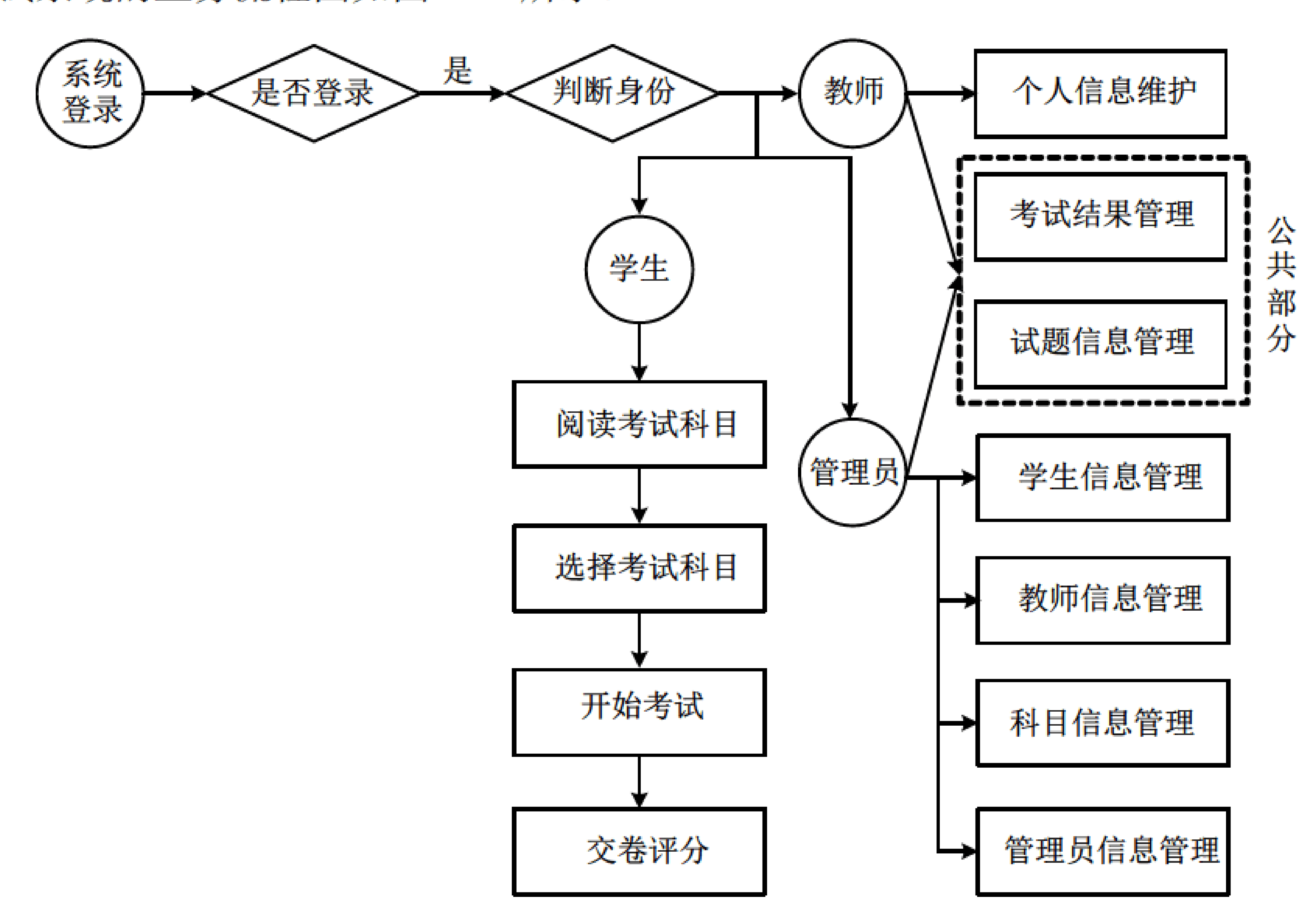


图 25.3　在线考试系统的业务流程图

25.3.4　构建开发环境

1．网站开发环境

☑　网站开发环境：Microsoft Visual Studio 2017 及以上。

☑　网站开发语言：ASP.NET+C#。

☑　网站后台数据库：SQL Server 2014。

☑　开发环境运行平台：Windows 7（SP1）/ Windows Server 8/Windows 10。



2．服务器端

☑　操作系统：Windows 7。

☑　Web 服务器：IIS 7.0 以上版本。

☑　数据库服务器：SQL Server 2014。

☑　网站服务器运行环境：Microsoft .NET Framework SDK v4.7。

3．客户端

☑　浏览器：Chrome 浏览器、Firefox 浏览器。

25.3.5　系统预览

在线考试系统由多个页面组成，下面仅列出几个典型页面，其他页面可参见资源包中的源程序。

考试界面如图 25.4 所示，主要实现考试系统的随机抽取试题、考生答卷、考试计时、限时自动交卷功能。后台管理员界面如图 25.5 所示，主要实现教师信息管理、管理员信息维护、题信息管理、考试科目信息管理以及考试结果管理。教师界面如图 25.6 所示，主要功能是教师对试题进行管理。考试评分界面如图 25.7 所示，主要功能是对考生答案进行评分。

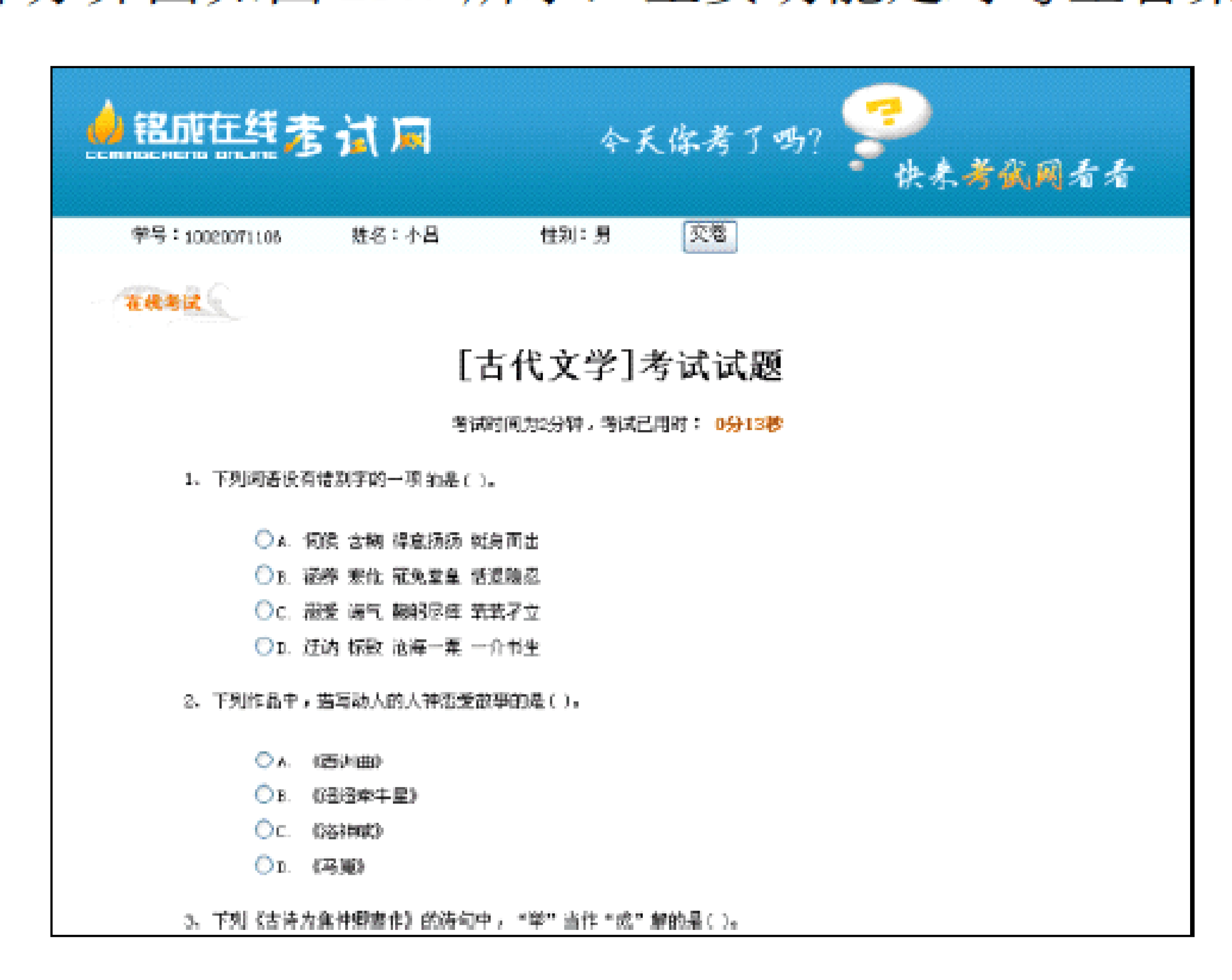


图 25.4　考试界面



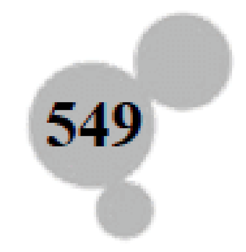
图 25.5　后台管理员界面



图 25.6　教师管理界面



图 25.7　考试评分界面



25.3.6　数据库设计

在开发在线考试系统之前，分析了系统的数据量，由于在线考试系统中试题及考生信息的数据量会很大，因此选择 Microsoft SQL Server 2014 数据库存储数据信息，数据库命名为 db_ExamOnline，在数据库中创建了 6 个数据表用于存储不同的信息，如图 25.8 所示。

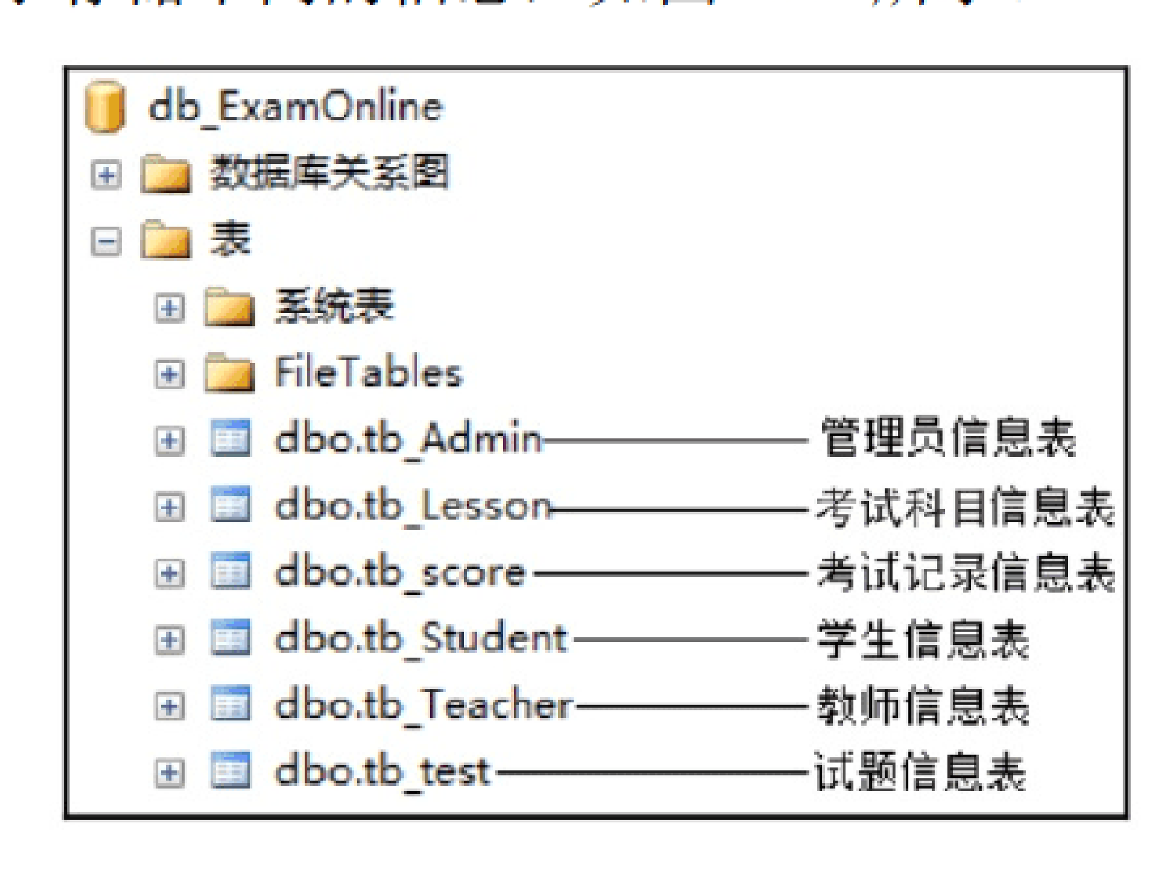


图 25.8　在线考试系统中用到的数据表

25.3.7　数据库概念设计

开发在线考试系统时，为了灵活地维护系统，设计了后台管理员模块，通过后台管理员模块可以方便地对整个在线考试系统进行维护，这时必须建立一个数据表用于存储所有的管理员信息。管理员信息实体 E-R 图如图 25.9 所示。

当考生成功登录在线考试系统后，可以根据需要选择考试科目，考生不同可能选择的考试科目会不同，系统必须提供一些参加考试的科目供考生选择，这时在数据库中应该建立一个存储所有参加考试科目的数据表。考试科目信息实体 E-R 图如图 25.10 所示。

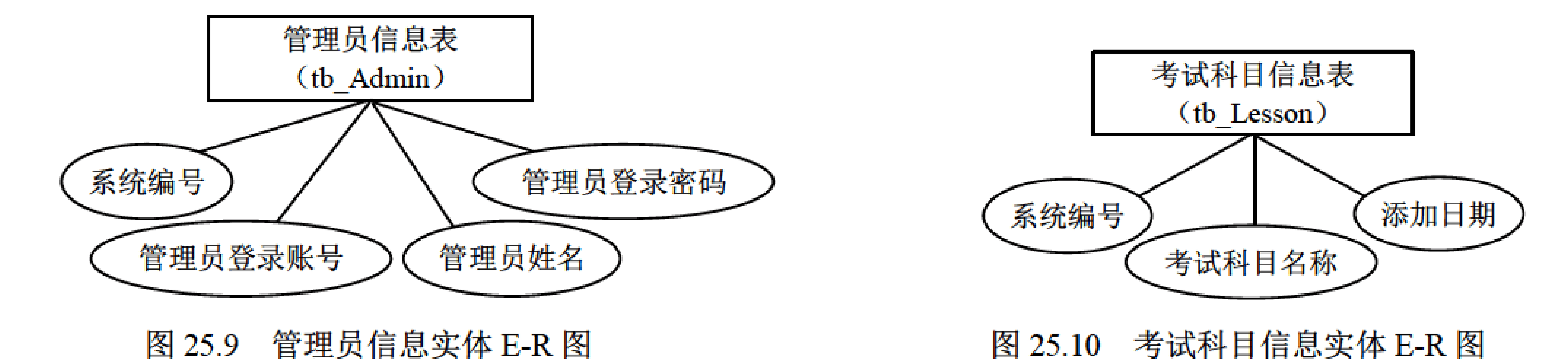


图 25.9　管理员信息实体 E-R 图　　　图 25.10　考试科目信息实体 E-R 图

考生选择考试科目，开始在线考试。在规定时间内必须完成考试，否则系统会自动提交试卷，并且将考生的考试成绩保存在数据表中。这样，方便后期查询考生是否参加过考试，以及查询历史考试得分。考试记录信息实体 E-R 图如图 25.11 所示。

在数据库中建立一个用于存储考生各项信息的数据表，其中包括考生登录时的账号（考生编号或考生学号）及密码。考生信息实体 E-R 图如图 25.12 所示。

为了方便教师对考试试题及考生考试结果进行管理，在数据库中必须建立一个数据表用于存储所



有的教师信息，其中包括教师登录后台管理系统时需要的账号及密码，以及教师负责的科目名称。教师信息实体 E-R 图如图 25.13 所示。

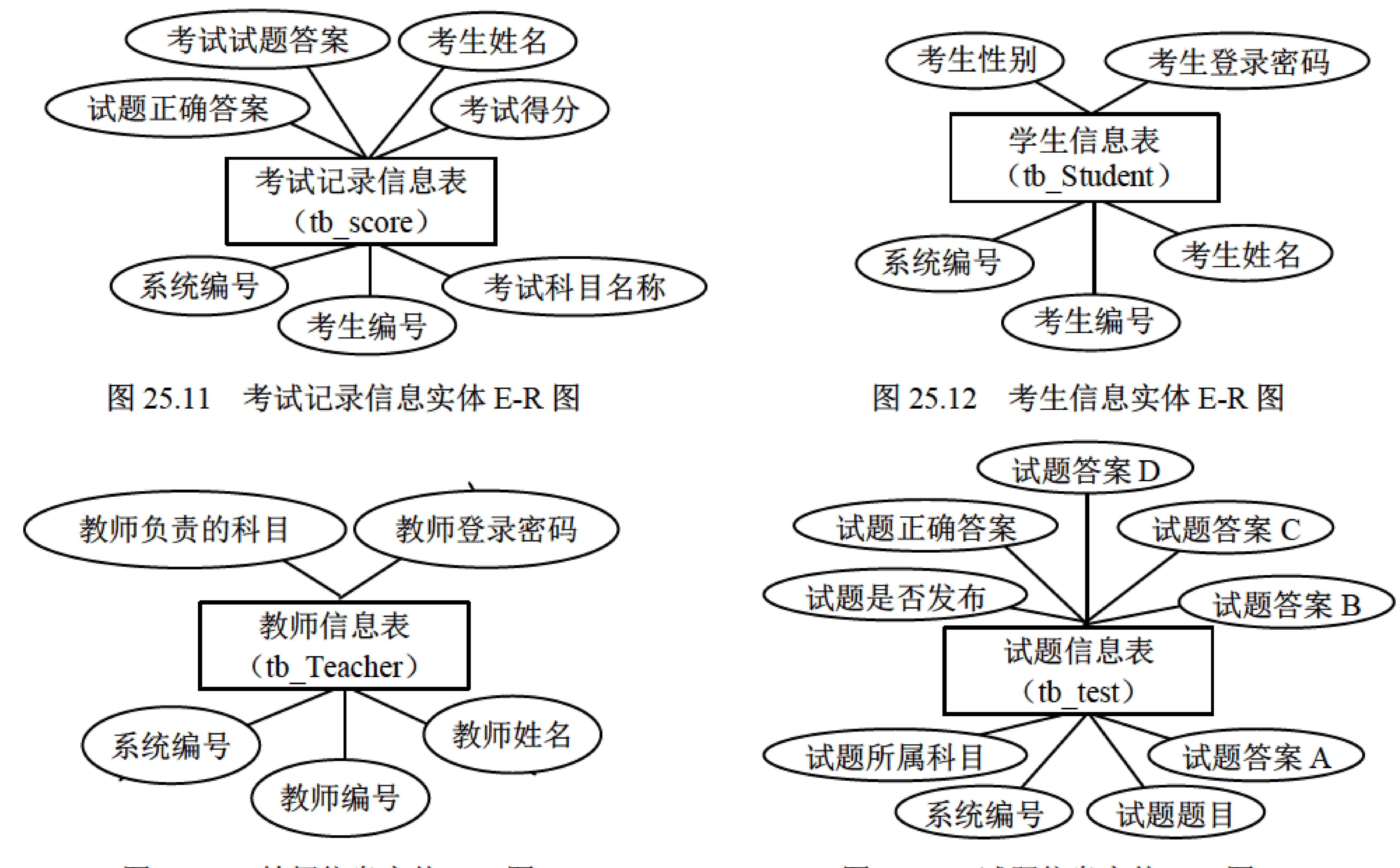


图 25.11　考试记录信息实体 E-R 图

图 25.12　考生信息实体 E-R 图

图 25.13　教师信息实体 E-R 图

图 25.14　试题信息实体 E-R 图

在线考试系统中考试试题是通过对数据库中存储的所有试题随机抽取产生的，所以必须在数据库中建立一个数据表用于存储所有参与考试的试题信息，其中包括试题题目、试题的 4 个备选答案、正确答案以及所属的科目。试题信息实体 E-R 图如图 25.14 所示。

25.3.8　数据库逻辑结构设计

根据设计好的 E-R 图在数据库中创建各表，系统数据库中各表的结构如下。

☑　管理员信息表

管理员信息表（tb_Admin）用于保存所有管理员信息，该表的结构如表 25.1 所示。

表 25.1　管理员信息表结构

字　段　名	数 据 类 型	长　　度	主　　键	描　　述
ID	int	4	是	系统编号
AdminNum	varchar	50	否	管理员编号
AdminName	varchar	50	否	管理员姓名
AdminPwd	varchar	50	否	管理员登录密码

☑　考试科目信息表

考试科目信息表（tb_Lesson）用于保存所有考试科目信息，该表的结构如表 25.2 所示。

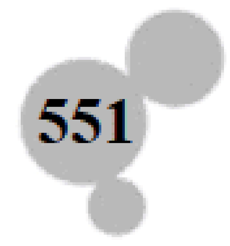


表 25.2　考试科目信息表结构

字 段 名	数 据 类 型	长　度	主　键	描　述
ID	int	4	是	系统编号
LessonName	varchar	50	否	考试科目名称
LessonDataTime	datetime	8	否	添加日期

☑　考试记录信息表

考试记录信息表（tb_score）用于保存所有参加过考试的考生的考试记录，该表的结构如表 25.3 所示。

表 25.3　考试记录信息表

字 段 名	数 据 类 型	长　度	主　键	描　述
ID	int	4	是	系统编号
StudentID	varchar	50	否	参加考试的考生编号
LessonName	varchar	50	否	考试科目名称
score	int	4	否	考生得分
StudentName	varchar	50	否	参加考试的考生姓名
StudentAns	varchar	50	否	考生试题答案
RightAns	varchar	50	否	试题正确答案

☑　学生信息表

学生信息表（tb_Student）用于保存所有考生信息，该表的结构如表 25.4 所示。

表 25.4　学生信息表

字 段 名	数 据 类 型	长　度	主　键	描　述
ID	int	4	是	系统编号
StudentNum	varchar	50	否	考生编号
StudentName	varchar	50	否	考生姓名
StudentPwd	varchar	50	否	考生登录密码
StudentSex	varchar	50	否	考生性别

☑　教师信息表

教师信息表（tb_Teacher）用于保存所有教师信息，该表的结构如表 25.5 所示。

表 25.5　教师信息表

字 段 名	数 据 类 型	长　度	主　键	描　述
ID	int	4	是	系统编号
TeacherNum	varchar	50	否	教师编号
TeacherName	varchar	50	否	教师姓名
TeacherPwd	varchar	50	否	教师登录密码
TeacherCourse	varchar	50	否	教师负责的科目

☑　试题信息表

试题信息表（tb_test）用于保存所有考试试题信息，该表的结构如表 25.6 所示。



表 25.6　试题信息表

字 段 名	数 据 类 型	长　度	主　键	描　述
ID	int	4	是	系统编号
testContent	varchar	200	否	试题题目
testAns1	varchar	50	否	试题备选答案 A
testAns2	varchar	50	否	试题备选答案 B
testAns3	varchar	50	否	试题备选答案 C
testAns4	varchar	50	否	试题备选答案 D
rightAns	varchar	50	否	试题正确答案
pub	int	4	否	试题是否发布
testCourse	varchar	50	否	试题所属科目

25.3.9　文件夹组织结构

每个网站都会有相应的文件夹组织结构，如果网站中网页数量很多，可以将所有的网页及资源放在不同的文件夹中。如果网站中网页不是很多，可以将图片、公共类或者程序资源文件放在相应的文件夹中，而网页可以直接放在网站根目录下。在线考试系统就是按照前者的文件夹组织结构排列的，如图 25.15 所示。

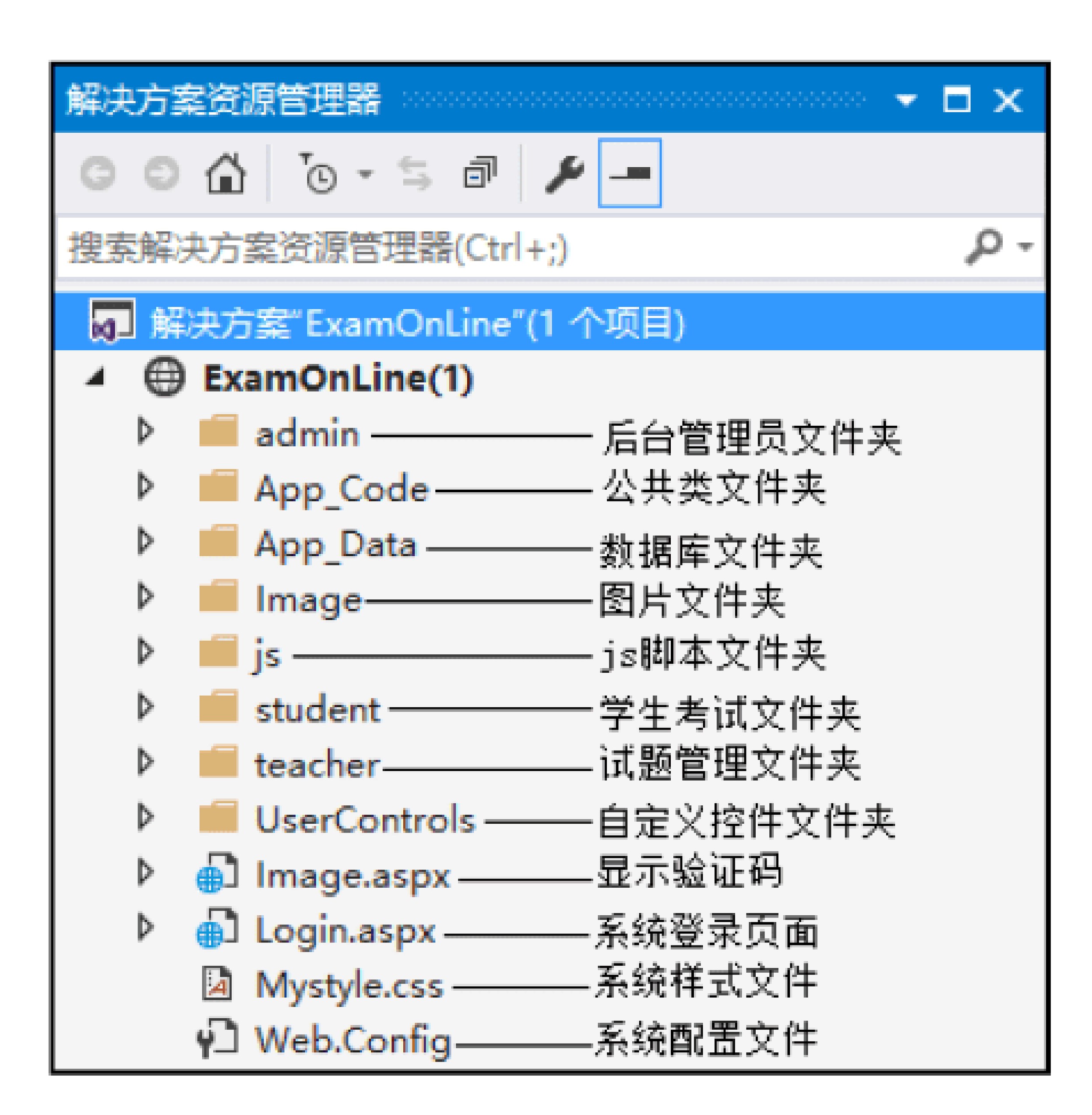


图 25.15　网站文件夹组织结构

25.4　公共类设计

在开发项目中以类的形式来组织、封装一些常用的方法和事件，不仅可以提高代码的重用率，也大大方便了代码的管理。本系统中创建了一个公共类 BaseClass，其中包含了 DBCon、BindDG、OperateData、CheckStudent、CheckTeacher 和 CheckAdmin 方法，分别用于连接数据库、绑定 GridView

控件、执行 SQL 语句、判断考生登录、判断教师登录和判断管理员登录。代码如下：

```
public class BaseClass
{
    public static SqlConnection DBCon()                                    //建立连接数据库的公共方法
    {
        return new SqlConnection("server=.;database=db_ExamOnline;uid=sa;pwd=");
    }
    public static void BindDG(GridView dg,string id, string strSql,string Tname)//建立绑定 GridView 控件的方法
    {
        SqlConnection conn = DBCon();                                      //连接数据库
        SqlDataAdapter sda = new SqlDataAdapter(strSql,conn);
        DataSet ds = new DataSet();
        sda.Fill(ds,Tname);
        dg.DataSource=ds.Tables[Tname];                                    //设置绑定数据源
        dg.DataKeyNames = new string[] { id };
        dg.DataBind();                                                     //绑定控件
    }
    public static void OperateData(string strsql)                          //建立一个执行 SQL 语句的方法
    {
        SqlConnection conn = DBCon();                                      //连接数据库
        conn.Open();                                                       //打开数据库
        SqlCommand cmd = new SqlCommand(strsql,conn);
        cmd.ExecuteNonQuery();
        conn.Close();                                                      //关闭连接
    }
    public static bool CheckStudent(string studentNum,string studentPwd)   //判断是否是学生登录
    {
        SqlConnection conn = DBCon();                                      //连接数据库
        conn.Open();                                                       //打开数据库
        SqlCommand cmd = new SqlCommand("select count(*) from tb_Student where StudentNum=
'"+studentNum+"' and StudentPwd='"+studentPwd+"'",conn);
        int i = Convert.ToInt32(cmd.ExecuteScalar());                      //返回值
        if (i > 0)                                                         //判断返回值是否大于 0
        {
            return true;                                                   //返回 true
        }
        else
        {
            return false;                                                  //返回 false
        }
        conn.Close();
    }
    public static bool CheckTeacher(string teacherNum, string teacherPwd)  //判断是否是教师登录
    {
        SqlConnection conn = DBCon();                                      //连接数据库
```

```
        conn.Open();                                                    //打开数据库
        SqlCommand cmd = new SqlCommand("select count(*) from tb_Teacher where TeacherNum='" +
teacherNum + "' and TeacherPwd='" + teacherPwd + "'", conn);
        int i = Convert.ToInt32(cmd.ExecuteScalar());                   //返回值
        if (i > 0)                                                      //判断返回值是否大于 0
        {
            return true;                                                //返回 true
        }
        else
        {
            return false;                                               //返回 false
        }
        conn.Close();                                                   //关闭连接
    }
    public static bool CheckAdmin(string adminNum, string adminPwd)     //判断是否是管理员登录
    {
        SqlConnection conn = DBCon();                                   //连接数据库
        conn.Open();                                                    //打开连接
        SqlCommand cmd = new SqlCommand("select count(*) from tb_Admin where AdminNum='" +
adminNum + "' and adminPwd='" + adminPwd + "'", conn);
        int i = Convert.ToInt32(cmd.ExecuteScalar());                   //返回值
        if (i > 0)                                                      //返回值是否大于 0
        {
            return true;                                                //返回 true
        }
        else
        {
            return false;                                               //返回 false
        }
        conn.Close();                                                   //关闭连接
    }
}
```

25.5　登录模块设计

25.5.1　登录模块概述

并不是任何人都可以参加在线考试，默认是不允许匿名登录的，只有经过管理员分配的编号和密码才能登录在线考试系统参加考试，这时就需要通过登录模块验证登录用户的合法性。登录模块是在线考试系统的第一道安全屏障，登录模块运行结果如图 25.16 所示。



图 25.16　登录模块运行结果

25.5.2　登录模块技术分析

登录模块中使用了验证码技术，通过验证码可以防止利用机器人软件反复自动登录。登录模块中的验证码主要是通过 Random 类实现的，为了更好地理解其用法，下面进行详细讲解。

Random 类表示伪随机数生成器，一种能够产生满足某些随机性统计要求的数字序列的设备，Random 类中最常用的是 Random.Next 方法。

Random.Next 方法用于返回一个指定范围内的随机数。其语法格式如下：

```
public virtual int Next (int minValue,int maxValue)
```

☑ minValue：返回随机数的下界。
☑ maxValue：返回随机数的上界，maxValue 必须大于或等于 minValue。
☑ 返回值：一个大于或等于 minValue 且小于 maxValue 的 32 位带符号整数，即返回值的范围包括 minValue 但不包括 maxValue。如果 minValue 等于 maxValue，则返回 minValue。

例如：

```
string MaxNum = "";                                    //建立上界变量
string MinNum = "";                                    //建立下界变量
for (int i = 0; i < 5; i++)
{
    MaxNum = MaxNum + "5";                             //设置上界
}
MinNum = MaxNum.Remove(0, 1);                          //设置下界
Random rd = new Random();                              //实例化 Random
string VNum = Convert.ToString(rd.Next(Convert.ToInt32(MinNum), Convert.ToInt32(MaxNum)));
return VNum;
```

25.5.3　登录模块实现过程

登录模块的具体实现步骤如下。

（1）新建一个网页，命名为 Login.aspx，主要用于实现系统的登录功能。该页面中用到的主要控件如表 25.7 所示。

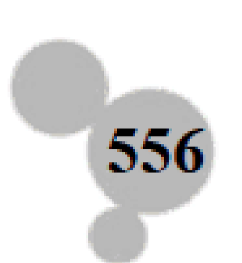


表 25.7　登录页面用到的主要控件

控件类型	控件 ID	主要属性设置	用　途
TextBox	txtNum	无	输入登录用户名
	txtPwd	TextModed 属性设置为 Password	输入登录用户密码
	txtCode	无	输入验证码
DropDownList	ddlstatus	Items 属性中添加 3 项	选择登录身份
Image	Image1	ImageUrl 属性设置为~/Image.aspx	显示验证码
Button	btnlogin	Text 属性设置为“登录”	登录
	btnconcel	Text 属性设置为“取消”	取消

（2）输入账号和密码等信息无误后，单击“登录”按钮进行登录。程序首先会判断输入的验证码是否正确，如果正确，则根据选择的登录身份调用公共类中相应的方法验证账号和密码是否正确，如果登录的账号和密码正确，则会转向与登录身份相符的页面。代码如下：

```
if (txtCode.Text.Trim() != Session["verify"].ToString())
 {
     Response.Write("<script>alert('验证码错误');location='Login.aspx'</script>");   //输入错误提示
 }
 else
 {
     if (this.ddlstatus.SelectedValue == "学生")                                    //如果登录身份为学生
     {
         if (BaseClass.CheckStudent(txtNum.Text.Trim(), txtPwd.Text.Trim()))        //验证登录账号和密码
         {
             Session["ID"] = txtNum.Text.Trim();
             Response.Redirect("student/studentexam.aspx");                         //转向考试界面
         }
         else
         {
             Response.Write("<script>alert('您不是学生或者用户名和密码错误');location='Login.aspx'</script>");
         }
     }
     if (this.ddlstatus.SelectedValue == "教师")                                    //如果登录身份为教师
     {
         if (BaseClass.CheckTeacher(txtNum.Text.Trim(), txtPwd.Text.Trim()))        //验证教师账号和密码
         {
             Session["teacher"] = txtNum.Text;
             Response.Redirect("teacher/TeacherManage.aspx");                       //转向试题管理模块
         }
         else
         {
             Response.Write("<script>alert('您不是教师或者用户名和密码错误');location='Login.aspx'</script>");
         }
     }
     if (this.ddlstatus.SelectedValue == "管理员")                                  //如果登录身份为管理员
     {
         if (BaseClass.CheckAdmin(txtNum.Text.Trim(), txtPwd.Text.Trim()))          //验证管理员账号和密码
         {
```

```
                Session["admin"] = txtNum.Text;
                Response.Redirect("admin/AdminManage.aspx");                   //转向后台管理员模块
            }
            else
            {
                Response.Write("<script>alert('您不是管理员或者用户名和密码错误');location='Login.aspx'</script>");
            }
        }
    }
```

（3）单击“取消”按钮，关闭登录窗口。代码如下：

```
protected void btnconcel_Click(object sender, EventArgs e)
{
    RegisterStartupScript("提示", "<script>window.close();</script>");
}
```

25.6 随机抽取试题模块设计

25.6.1 随机抽取试题模块概述

开发在线考试系统过程中，需要考虑的一点是如何将试题显示在页面上，如何将试题从数据库中读取出来。比较合理的做法是将所有试题信息存储在数据库中，然后随机抽取若干道试题，动态地显示在页面中。为了实现此功能，设计出随机抽取试题模块，运行结果如图 25.17 所示。

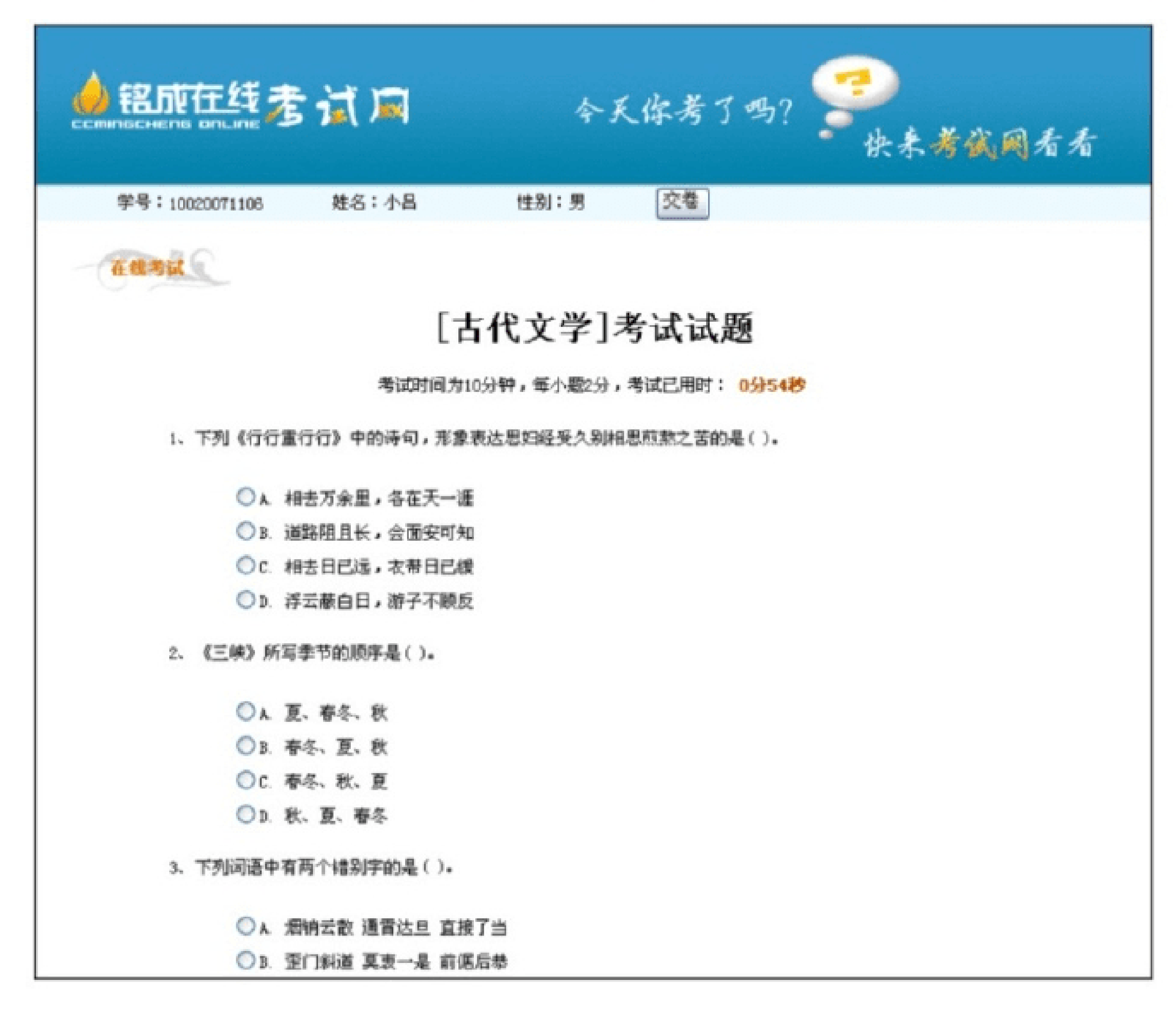
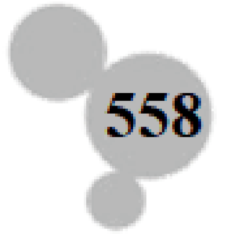


图 25.17 随机抽取考试试题



25.6.2　随机抽取试题模块技术分析

实现随机抽取试题模块的关键技术是 SQL Server 中的 NEWID 函数，通过此函数可以动态创建 uniqueidentifier 类型的值，即随机数，实现起来非常简单。有关 NEWID 函数的详细说明如下。

NEWID 函数的功能是创建 uniqueidentifier 类型的唯一值。其语法格式如下：

```
NEWID( )
```

返回类型：uniqueidentifier。

例如，对变量使用 NEWID 函数，使用 NEWID 对声明为 uniqueidentifier 数据类型的变量赋值。在测试该值前，将先打印 uniqueidentifier 数据类型变量的值。

```
-- Creating a local variable with DECLARE/SET syntax.
DECLARE @myid uniqueidentifier
SET @myid = NEWID()
PRINT 'Value of @myid is: '+ CONVERT(varchar(255), @myid)
```

下面是结果集：

```
Value of @myid is: 6F9619FF-8B86-D011-B42D-00C04FC964FF
```

例如，从数据表 tb_Test 中随机抽取 10 条数据，可以利用下面的代码实现：

```
Select top 10 * from tb_Test order by newid()
```

25.6.3　随机抽取试题模块实现过程

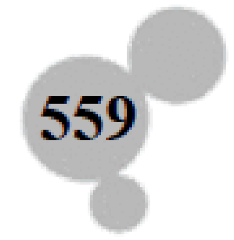
随机抽取试题模块的具体实现步骤如下。

（1）在随机抽取试题之前，考生要选择考试的科目，然后根据选择的科目随机从数据库中抽取试题给考生。所以，考生选择考试科目是随机抽取试题的条件，其运行结果如图 25.18 所示。



图 25.18　选择考试科目

程序首先根据考生选择的科目对数据库进行检索，查看数据库中是否有相关的试题。如果存在试题，则跳转到随机抽取试题页面；否则，提示考生选择的考试科目在数据库中没有试题。代码如下：



```
protected void Button2_Click(object sender, EventArgs e)
 {
        string StuID = Session["ID"].ToString();                                           //考生的编号
        string StuKC = ddlKm.SelectedItem.Text;                                            //选择的考试科目
        SqlConnection conn = BaseClass.DBCon();                                            //连接数据库
        conn.Open();                                                                       //打开连接
        SqlCommand cmd = new SqlCommand("select count(*) from tb_score where StudentID=
'" + StuID + "' and LessonName='" + StuKC + "'", conn);                                    //执行 SQL 语句
        int i = Convert.ToInt32(cmd.ExecuteScalar());                                      //获取返回值
        if (i > 0)                                                                         //如果返回值大于 0
        {
            MessageBox.Show("你已经参加过此科目的考试了");
        }
        else
        {
            cmd = new SqlCommand("select count(*) from tb_test where testCourse='"+StuKC+"'", conn);
            int N = Convert.ToInt32(cmd.ExecuteScalar());                                  //获取返回值
            if (N >0)                                                                      //如果返回值大于 0
            {
                cmd = new SqlCommand("insert into tb_score(StudentID,LessonName,StudentName)
values('" + StuID "','" + tuKC + "','" + lblName.Text + "')", conn);                        //执行 SQL 语句
                cmd.ExecuteNonQuery();
                conn.Close();                                                              //关闭连接
                Session["KM"] = StuKC;
                Response.Write("<script>window.open('StartExam.aspx','newwindow','status=
                    1,scrollbars= 1,resizable=1')</script>");
                Response.Write("<script>window.opener=null;window.close();</script>");
            }
            else
            {
                MessageBox.Show("此科目没有考试题");                                        //弹出提示信息
                return;
            }
        }
 }
```

（2）新建一个网页，命名为 StartExam.aspx，作为随机抽取试题页面及考试页面。该页面中用到的主要控件如表 25.8 所示。

表 25.8　随机抽取试题页面用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　　途
A Label	lblStuNum	无	显示考生编号
	lblStuName	无	显示考生姓名
	lblStuSex	无	显示考生性别
	lblStuKM	无	显示考试科目
	lblEndtime	无	显示考试声明
	lbltime	无	显示考试用时时间

续表

控 件 类 型	控件 ID	主要属性设置	用　　途
Panel	Panel1	无	显示随机抽取的试题
Button	btnsubmit	无	提交试卷

（3）当页面加载时，根据考生选择的科目在数据库中随机抽取试题，并显示在 Panel 控件中。代码如下：

```
public string Ans = null;                                             //建立存储正确答案的公共变量
public int tNUM;                                                      //记录考题数量
protected void Page_Load(object sender, EventArgs e)
{
    lblEndtime.Text = "考试时间为 10 分钟，每小题 2 分，考试已用时：";        //显示考试提示
    lblStuNum.Text = Session["ID"].ToString();                        //显示考生编号
    lblStuName.Text = Session["name"].ToString();                     //显示考生姓名
    lblStuSex.Text = Session["sex"].ToString();                       //显示考生性别
    lblStuKM.Text = "[" + Session["KM"].ToString() + "]" + "考试试题";    //显示考试科目
    int i=1;                                                          //初始化变量
    SqlConnection conn = BaseClass.DBCon();                           //连接数据库
    conn.Open();                                                      //打开连接
    SqlCommand cmd = new SqlCommand("select top 10 * from tb_test where testCourse='" + Session["KM"].
ToString() + "' order by newid()", conn);
     SqlDataReader sdr = cmd.ExecuteReader();                         //创建记录集
    while (sdr.Read())
    {
        Literal littxt = new Literal();                               //创建 Literal 控件
        Literal litti = new Literal();                                //创建 Literal 控件
        RadioButtonList cbk = new RadioButtonList();                  //创建 RadioButtonList 控件
        cbk.ID = "cbk" + i.ToString();
        littxt.Text = i.ToString() + "、" + Server.HtmlEncode(sdr["testContent"].ToString()) + "<br>ckquote>";
        litti.Text = "</Blockquote>";
        cbk.Items.Add("A. " + Server.HtmlEncode(sdr["testAns1"].ToString()));//添加选项 A
        cbk.Items.Add("B. " + Server.HtmlEncode(sdr["testAns2"].ToString()));//添加选项 B
        cbk.Items.Add("C. " + Server.HtmlEncode(sdr["testAns3"].ToString()));//添加选项 C
        cbk.Items.Add("D. " + Server.HtmlEncode(sdr["testAns4"].ToString()));//添加选项 D
        cbk.Font.Size = 11;                                           //设置文字大小
        for (int j = 1; j <= 4; j++)
        {
            cbk.Items[j - 1].Value = j.ToString();
        }
        Ans += sdr[6].ToString();                                     //获取试题的正确答案
        if (Session["a"] == null)                                     //判断是否第一次加载
        {
           //如果第一次加载则将正确答案赋值给 Session["Ans"]
            Session["Ans"] = Ans;
         }
```

```
            Panel1.Controls.Add(littxt);                                    //将控件添加到 Panel 中
            Panel1.Controls.Add(cbk);                                       //将控件添加到 Panel 中
            Panel1.Controls.Add(litti);                                     //将控件添加到 Panel 中
            i++;                                                            //使 i 递增
            tNUM++;                                                         //使 tNUM 递增
        }
        sdr.Close();
        conn. Close();                                                      //关闭连接
        Session["a"] = 1;
}
```

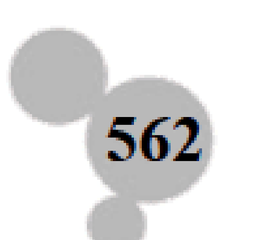
（4）考生在规定的时间内进行考试，当考生答题完毕，单击“交卷”按钮提交试卷，此时系统会将该考生的答题结果提交给自动评分模块。代码如下：

```
protected void btnsubmit_Click(object sender, EventArgs e)
{
    string msc = "";                                                        //建立变量 msc 存储考生答案
    for (int i = 1; i <= 10; i++)
    {
        RadioButtonList list = (RadioButtonList)Panel1.FindControl("cbk" + i.ToString());
        if (list != null)
        {
            if (list.SelectedValue.ToString() != "")
                msc += list.SelectedValue.ToString();                       //存储考生答案
            else
                msc += "0";                                                 //如果没有选择则为 0
        }
    }
    Session["Sans"] = msc;                                                  //考生答案
    //更新考试记录数据表
    string sql = "update tb_score set RigthAns='" + Ans + "' where StudentID='" + lblStuNum.Text + "'";
    BaseClass.OperateData(sql);
     //更新考试记录数据表
    string strsql = "update tb_score set StudentAns='" + msc + "' where StudentID='" + lblStuNum.Text + "'";
    BaseClass.OperateData(strsql);
    Response.Redirect("result.aspx?BInt=" + tNUM.ToString());
}
```

25.7　自动评分模块设计

25.7.1　自动评分模块概述

在线考试系统和普通考试的流程是一样的，考生答卷完毕后要对考生的答案评分。根据实际需要，



在线考试系统中加入了自动评分模块，当考生答题完毕提交试卷时，系统会根据考生选择的答案与正确答案进行比较，最后进行评分，运行结果如图 25.19 所示。



图 25.19　自动评分模块运行结果

25.7.2　自动评分模块技术分析

自动评分模块使用的基本技术是字符串的截取与比较，下面介绍使用 Substring 和 Equals 方法对字符串进行截取与比较。

1．截取字符串

使用 Substring 方法可以从指定字符串中截取子串。语法格式如下：

```
public string Substring(int startIndex,int length)
```

☑　startIndex：子字符串的起始位置的索引。

☑　length：子字符串中的字符数。

例如，将字符串“我们是社会主义新青年”截取为“社会主义新青年”。代码如下：

```
string str = "我们是社会主义新青年";
string str2 = str.Substring(3,str.Length-3);
Response.Write(str2);
```

2．比较字符串

Equals 方法用于确定两个 String 对象是否具有相同的值。语法格式如下：

```
public bool Equals(string value)
```

例如，判断字符串 stra 和字符串 strb 是否相等。代码如下：

```
stra.Equals(strb)
```

如果 stra 的值与 strb 相同，则为 true；否则为 false。

25.7.3　自动评分模块实现过程

自动评分模块的具体实现步骤如下。

（1）新建一个网页，命名为 result.aspx，主要用于实现对考生提交的试题答案进行自动评分。该页面中用到的主要控件如表 25.9 所示。

表 25.9　自动评分页面用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　　途
A Label	lbldate	无	显示当前系统时间
	lblkm	无	显示考生考试科目
	lblnum	无	显示考生编号
	lblname	无	显示考生姓名
	lblResult	无	显示考试得分

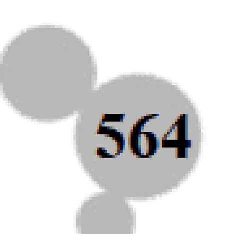
（2）考生将试题答案提交到自动评分模块，自动评分模块对考生答案进行评分，并将考生的成绩添加到数据表 tb_score 中。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    string Rans = Session["Ans"].ToString();                    //获取正确答案
    int j = Convert.ToInt32(Request.QueryString["BInt"]);       //获取试题数量
    string Sans = Session["Sans"].ToString();                   //获取考生答案
    int StuScore = 0;                                           //将考试成绩初始化为 0
    for (int i = 0; i < j; i++)
    {
        if (Rans.Substring(i, 1).Equals(Sans.Substring(i, 1)))  //将考生答案与正确答案进行比较
        {
            StuScore += 2;                                      //如果答案正确加 2 分
        }
    }
    this.lblResult.Text = StuScore.ToString();                  //显示考试成绩
    this.lblkm.Text = Session["KM"].ToString();                 //显示考试科目
    this.lblnum.Text = Session["ID"].ToString();                //显示考生编号
    this.lblname.Text = Session["name"].ToString();             //显示考生姓名
    //更新考试记录数据表
    string strsql = "update tb_score set score='" + StuScore.ToString() + "' where StudentID='" + Session["ID"].
ToString() + "' and LessonName='" + Session["KM"].ToString() + "'";
    BaseClass.OperateData(strsql);
}
```

25.8　教师管理模块设计

25.8.1　教师管理模块概述

教师管理模块在整个在线考试系统中占有非常重要的地位，它是专门为教师设计的。教师登录此



模块后即可在后台对试题进行添加、修改和删除，并且可以查看考试结果。教师管理模块运行结果如图 25.20 所示。



图 25.20　教师管理模块运行结果

25.8.2　教师管理模块技术分析

在开发教师管理模块时，主要应用了对数据库进行查询、添加、更新、删除以及模糊查询等技术，下面主要对模糊查询进行介绍。

在进行数据查询时，经常会使用模糊查询方式。模糊查询是指根据输入的条件进行模式匹配，即将输入的查询条件按照指定的通配符与数据表中的数据进行匹配，查找符合条件的数据。模糊查询一般应用在不能准确写出查询条件的情况。在设计模糊查询时，一般通过文本框获取查询条件，这样可以使查询更为灵活。模糊查询通常使用 LIKE 关键字来指定模式查询条件。

LIKE 关键字的语法格式如下：

```
match_expression [ NOT ] LIKE pattern [ ESCAPE escape_character ]
```

- ☑ match_expression：任何字符串数据类型的有效 SQL Server 表达式。
- ☑ pattern：match_expression 中的搜索模式。
- ☑ escape_character：字符串数据类型分类中的所有数据类型的任何有效 SQL Server 表达式。escape_character 没有默认值，且必须仅包含一个字符。

LIKE 查询条件需要使用通配符在字符串内查找指定的模式，LIKE 关键字中的通配符如表 25.10 所示。

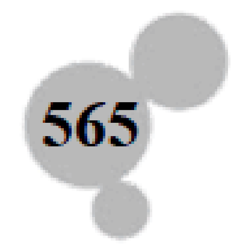


表 25.10　LIKE 关键字中的通配符及其含义

通　配　符	说　　明
%	由 0 个或更多字符组成的任意字符串
	任意单个字符
[ ]	用于指定范围，例如[A～F]，表示 A～F 范围内的任何单个字符
[^]	表示指定范围之外的，例如[^ A～F]，表示 A～F 范围以外的任何单个字符

1．“%”通配符

“%”通配符能匹配 0 个或更多个字符的任意长度的字符串。

在 SQL Server 语句中，可以在查询条件的任意位置放置一个“%”符号来代表任意长度的字符串。在设置查询条件时，也可以放置两个“%”，但是最好不要连续出现两个“%”符号。

2．“_”通配符

“_”号表示任意单个字符，该符号只能匹配一个字符，利用“_”号可以作为通配符组成匹配模式进行查询。

“_”符号可以放在查询条件的任意位置，且只能代表一个字符。

3．“[ ]”通配符

在模式查询中可以使用“[ ]”符号来查询一定范围内的数据。“[ ]”符号用于表示一定范围内的任意单个字符，它包括两端数据。

4．“[^]”通配符

在模式查询中可以使用“[^]”符号来查询不在指定范围内的数据。“[^]”符号用于表示不在某范围内的任意单个字符，它包括两端数据。

25.8.3　教师管理模块实现过程

教师管理模块中具体包括试题基本信息、添加试题信息、考试结果和修改密码的功能。具体实现步骤如下。

教师通过登录模块成功登录后，系统会根据登录的账号对数据库进行检索，查找出该名教师的姓名和负责的课程。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["teacher"] == null)                          //禁止匿名登录
    {
        Response.Redirect("../Login.aspx");
    }
    else
    {
        lblwz.Text = Session["teacher"].ToString();          //教师编号
        SqlConnection conn = BaseClass.DBCon();              //连接数据库
        conn.Open();                                         //打开连接
```



```
            SqlCommand cmd = new SqlCommand("select * from tb_Teacher where TeacherNum='" + lblwz.Text + "'", conn);
            SqlDataReader sdr = cmd.ExecuteReader();                    //创建记录集
            sdr.Read();
            lblname.Text = sdr["TeacherName"].ToString();               //显示教师姓名
            int id = Convert.ToInt32(sdr["TeacherCourse"].ToString());  //获取教师的授课编号
            sdr.Close();
            cmd = new SqlCommand("select LessonName from tb_Lesson where ID="+id, conn);
            lblkc.Text = cmd.ExecuteScalar().ToString();                //获取教师授课科目名称
            Session["KCname"] = lblkc.Text;
            conn.Close();                                               //关闭连接
        }
}
```

1．试题基本信息（TExaminationInfo.aspx）

新建一个网页，命名为 TExaminationInfo.aspx，主要用于实现浏览所有的试题信息。该页面中用到的主要控件如表 25.11 所示。

表 25.11　试题基本信息页面中用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　　途
TextBox	txtstkey	无	输入查询关键字
Button	btnserch	Text 属性设置为“查询”	查询
GridView	gvExaminationInfo	Columns 属性中添加 4 列	显示所有试题信息及查询结果

当此页面加载时，从数据库中检索出所有的试题信息，显示在 GridView 控件上。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["teacher"] == null)                                     //禁止匿名登录
    {
        Response.Redirect("../Login.aspx");
    }
    else
    {
        if (!IsPostBack)
        {
            string strsql = "select * from tb_test where testCourse='" + Session["KCname"].ToString() + "'";
            BaseClass.BindDG(gvExaminationInfo, "ID", strsql, "ExaminationInfo");
        }
    }
}
```

在 GridView 控件的 RowDeleting 事件中添加代码，执行对指定数据的删除操作。代码如下：

```
protected void gvExaminationInfo_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    int id = (int)gvExaminationInfo.DataKeys[e.RowIndex].Value;        //获取欲删除信息的编号
    string sql = "delete from tb_test where ID=" + id;                  //执行删除操作的 SQL 语句
```

```
        BaseClass.OperateData(sql);
        string strsql = "select * from tb_test where testCourse='" + Session["KCname"].ToString() + "'";
        BaseClass.BindDG(gvExaminationInfo, "ID", strsql, "ExaminationInfo");
}
```

对 GridView 控件进行分页，要在其 PageIndexChanging 中添加分页绑定代码，才能在分页时正常显示数据。代码如下：

```
protected void gvExaminationInfo_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
        gvExaminationInfo.PageIndex = e.NewPageIndex;
        string strsql = "select * from tb_test where testCourse='" + Session["KCname"].ToString() + "'";
        BaseClass.BindDG(gvExaminationInfo, "ID", strsql, "ExaminationInfo");
}
```

当在“关键字”文本框中输入查询的关键字之后，单击“查询”按钮查询与关键字相关的数据。代码如下：

```
protected void btnserch_Click(object sender, EventArgs e)
{
        string strsql = "select * from tb_test where testContent like '%"+txtstkey.Text.Trim()+"%'";
        BaseClass.BindDG(gvExaminationInfo, "ID", strsql, "ExaminationInfo");
}
```

2．添加试题信息（TAddExamination.aspx）

新建一个网页，命名为 TAddExamination.aspx，主要用于实现添加试题信息。该页面中用到的主要控件如表 25.12 所示。

表 25.12　添加试题信息页面中用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　　途
TextBox	txtsubject	TextMode 属性设置为 MultiLine	输入试题题目
	txtAnsA	TextMode 属性设置为 MultiLine	输入答案选项 A
	txtAnsB	TextMode 属性设置为 MultiLine	输入答案选项 B
	txtAnsC	TextMode 属性设置为 MultiLine	输入答案选项 C
	txtAnsD	TextMode 属性设置为 MultiLine	输入答案选项 D
Button	btnconfirm	Text 属性设置为“确定”	确定
	btnconcel	Text 属性设置为“重置”	重置
RadioButtonList	rblRightAns	Items 属性中添加 4 项	选择正确答案
CheckBox	cbFB	Text 属性设置为“是否发布”	设置是否发布
Label	lblkmname	无	显示教师负责的课程

试题的所有信息输入完毕之后，单击“确定”按钮添加到数据库中。代码如下：

```
protected void btnconfirm_Click(object sender, EventArgs e)
{
        //判断信息填写是否完整
```

```
    if (txtsubject.Text == "" || txtAnsA.Text == "" || txtAnsB.Text == "" || txtAnsC.Text == "" || txtAnsD.Text == "" )
    {
        MessageBox.Show("请将信息填写完整");                    //弹出提示信息
        return;
    }
    else
    {
        string isfb = "";                                       //建立变量
        if (cbFB.Checked == true)                               //判断是否选择
            isfb = "1";                                         //如果选择赋值为 1
        else
            isfb = "0";                                         //否则赋值为 0
        string str = "insert into tb_testContent,testAns1,testAns2,testAns3,testAns4,rightAns,pub,testCourse)
values('" + txtsubject.Text.Trim() + "','" + nsA.Text.Trim() + "','" + txtAnsB.Text.Trim() + "','" + txtAnsC.Text.Trim()
+ "','" + txtAnsD.Text.Trim() + "','" + rblRigs.SelectedValue.ToString() + "','" + isfb + "','" +
Session["KCname"].ToString()+ "')";
        BaseClass.OperateData(str);                             //将数据插入数据库
        btnconcel_Click(sender, e);                             //清空所有输入的信息
    }
}
```

3．考试结果（TExaminationResult.aspx）

新建一个网页，命名为 TExaminationResult.aspx，主要用于实现浏览所有考生考试记录。该页面中用到的主要控件如表 25.13 所示。

表 25.13　考试结果页面中用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　　途
TextBox	txtkey	无	输入查询关键字
Button	btnserch	Text 属性设置为“查询”	查询
GridView	gvExaminationresult	Columns 属性中添加 5 列	显示所有考生考试结果
DropDownList	ddltype	Items 属性中添加两项	选择查询的范围

选择查询范围，输入查询关键字，单击“查询”按钮查询与关键字相关的信息，并显示在 GridView 控件上。代码如下：

```
protected void btnserch_Click(object sender, EventArgs e)
{
    string type = ddltype.SelectedItem.Text;                            //获取查询的范围
    if (type == "学号")                                                 //如果选择“学号”
    {
        string resultstr = "select * from tb_score where StudentID like '%" + txtkey.Text.Trim() + "%' and
LessonName ='" + Session ["KCname"]. ToString() + "'";
        BaseClass.BindDG(gvExaminationresult, "ID", resultstr, "result");   //在学号范围内查找
        Session["num"] = "学号";
    }
    if (type == "姓名")                                                 //如果选择“姓名”
```

```
    {
        string resultstr = "select * from tb_score where StudentName like '%" + txtkey.Text.Trim() + "%' and 
LessonName='" + Session["KCname"].ToString() + "'";
        BaseClass.BindDG(gvExaminationresult, "ID", resultstr, "result");       //在姓名范围内查找
        Session["num"] = "姓名";
    }
}
```

单击“删除”按钮可以删除指定的信息，在 GridView 控件的 RowDeleting 事件中添加如下代码：

```
protected void gvExaminationInfo_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    int id = (int)gvExaminationresult.DataKeys[e.RowIndex].Value;              //获取欲删除信息的 id
    string strsql = "delete from tb_score where ID=" + id;                     //执行删除操作的 SQL 语句
    BaseClass.OperateData(strsql);
    if (Session["num"].ToString() == "学号")                                    //判断当前查询的范围
    {
        string resultstr = "select * from tb_score where StudentID like '%" + txtkey.Text.Trim() + "%' and 
LessonName='" + Session["KCname"].ToString() + "'";
        BaseClass.BindDG(gvExaminationresult, "ID", resultstr, "result");       //绑定控件
    }
    else
    {
        string resultstr = "select * from tb_score where StudentName like '%" + txtkey.Text.Trim() + "%' and 
LessonName='" + Session["KCname"].ToString() + "'";
        BaseClass.BindDG(gvExaminationresult, "ID", resultstr, "result");       //绑定控件
    }
}
```

如果查询出的数据过多，可以对数据进行分页绑定，具体方法是在 GridView 控件的 PageIndex Changing 事件中添加如下代码：

```
protected void gvExaminationresult_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    if (Session["num"].ToString() == "学号")                                    //判断当前查询范围
    {
        gvExaminationresult.PageIndex = e.NewPageIndex;
        string resultstr = "select * from tb_score where StudentID like '%" + txtkey.Text.Trim() + "%' and 
LessonName='" + Session["KCname"].ToString() + "'";
        BaseClass.BindDG(gvExaminationresult, "ID", resultstr, "result");       //绑定控件
    }
    else
    {
        gvExaminationresult.PageIndex = e.NewPageIndex;
        string resultstr = "select * from tb_score where StudentName like '%" + txtkey.Text.Trim() + "%' and 
LessonName='" + Session["KCname"].ToString() + "'";
        BaseClass.BindDG(gvExaminationresult, "ID", resultstr, "result");       //绑定控件
```

```
    }
}
```

4．修改密码（TeacherChangePwd.aspx）

新建一个网页，命名为 TeacherChangePwd.aspx，主要用于实现教师修改密码。该页面中用到的主要控件如表 25.14 所示。

表 25.14　修改密码页面中用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　　途
TextBox	txtOldPwd	无	输入旧密码
	txtNewPwd	无	输入新密码
	txtNewPwdA	无	再次输入新密码
Button	btnchange	Text 属性设置为“确定修改”	确定修改

所有数据输入完毕后，单击“确定修改”按钮完成密码的修改。代码如下：

```
protected void btnchange_Click(object sender, EventArgs e)
{
    if (txtNewPwd.Text == "" || txtNewPwdA.Text == "" || txtOldPwd.Text == "") //检查信息输入是否完整
    {
        MessageBox.Show("请将信息填写完整");                              //弹出提示信息
        return;
    }
    else
    {
        //检查旧密码输入是否正确
        if (BaseClass.CheckTeacher(Session["teacher"].ToString(), txtOldPwd.Text.Trim()))
        {
            if (txtNewPwd.Text.Trim() != txtNewPwdA.Text.Trim())          //检查两次输入的新密码是否相等
            {
                MessageBox.Show("两次密码不一致");                          //弹出提示信息
                return;
            }
            else
            {
                string strsql = "update tb_Teacher set TeacherPwd='" + txtNewPwdA.Text.Trim() + "' where
TeacherNum='" + Session["teacher"].ToString() + "'";
                BaseClass.OperateData(strsql);                            //更新数据表
                MessageBox.Show("密码修改成功");
                txtNewPwd.Text = "";                                      //清空文本框
                txtNewPwdA.Text = "";                                     //清空文本框
                txtOldPwd.Text = "";                                      //清空文本框
            }
        }
        else
        {
```

```
                MessageBox.Show("旧密码输入错误");                         //弹出提示信息
                return;
            }
        }
}
if (!IsPostBack)
{
        string strsql = "select * from tb_test where testCourse='" + Session["KCname"].ToString() + "'";
        BaseClass.BindDG(gvExaminationInfo, "ID", strsql, "ExaminationInfo");
}
```

25.9　后台管理员模块设计

25.9.1　后台管理员模块概述

在线考试系统中，后台管理员模块具有最高权限，管理员通过登录模块成功登录后台管理员模块之后，可以对教师信息、学生信息、管理员信息、试题信息、考试科目信息以及考试结果进行管理，使系统维护起来更方便、快捷。后台管理员模块运行结果如图 25.21 所示。



图 25.21　后台管理员模块运行结果

25.9.2　后台管理员模块技术分析

在开发后台管理员模块过程中，使用比较频繁的是使用 Eval 方法绑定数据。Eval 方法是一个静态



方法，只能绑定到模板中的子控件的公共属性上。

Eval 方法的功能是将数据绑定到控件。其语法格式如下：

```
public static Object Eval(Object container,string expression)
```

☑ container：表达式根据其进行计算的对象引用。此标识符必须是以页的指定语言表示的有效对象标识符。
☑ expression：从 container 到要放置在绑定控件属性中的公共属性值的导航路径。此路径必须是以点分隔的属性或字段名称字符串。
☑ 返回值：Object 是数据绑定表达式的计算结果。

例如，将字段名为 Price 中的数据绑定到控件上，可以使用下面的代码实现：

```
<%# DataBinder.Eval(Container.DataItem, "Price") %>
```

25.9.3　后台管理员模块实现过程

后台管理员模块实现的具体功能有管理学生基本信息、添加学生信息、管理教师基本信息、添加教师信息、试题基本信息管理、添加试题信息、考试科目设置、查询考试结果以及管理员信息维护。具体的实现步骤如下。

1．管理学生基本信息（StudentInfo.aspx）

新建一个网页，命名为 StudentInfo.aspx，主要用于实现对学生基本信息的查询、修改和删除。该页面中用到的主要控件如表 25.15 所示。

表 25.15　管理学生基本信息页面中用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　　途
TextBox	txtKey	无	输入查询关键字
Button	btnserch	Text 属性设置为“查看”	查询
GridView	gvStuInfo	Columns 属性中添加 6 列	显示所有学生信息
DropDownList	ddlType	Items 属性中添加两项	选择查询的范围

当此页面加载时，首先绑定 GridView 控件，显示所有学生信息。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] == null)                                    //禁止匿名登录
    {
        Response.Redirect("../Login.aspx");
    }
    if (!IsPostBack)
    {
        string strsql = "select * from tb_Student order by ID desc";    //检索所有学生信息
        BaseClass.BindDG(gvStuInfo,"ID", strsql,"stuinfo");            //绑定控件
    }
}
```

要想查询学生信息，首先选择查询范围，然后在文本框中输入关键字，单击“查看”按钮进行查询。代码如下：

```
protected void btnserch_Click(object sender, EventArgs e)
{
    if (txtKey.Text == "")                                                  //检查是否输入了关键字
    {
        string strsql = "select * from tb_Student order by ID desc";        //检索所有学生信息
        BaseClass.BindDG(gvStuInfo, "ID", strsql, "stuinfo");               //绑定控件
    }
    else
    {
        string stype = ddlType.SelectedItem.Text;                           //获取查询范围
        string strsql = "";
        switch (stype)
        {
            case "学号":                                                     //如果查询范围是“学号”
                strsql = "select * from tb_Student where StudentNum like '%" + txtKey.Text.Trim() + "%'";
                BaseClass.BindDG(gvStuInfo, "ID", strsql, "stuinfo"); ;
                break;
            case "姓名":                                                     //如果查询范围是“姓名”
                strsql = "select * from tb_Student where StudentName like '%" + txtKey.Text.Trim() + "%'";
                BaseClass.BindDG(gvStuInfo, "ID", strsql, "stuinfo");
                break;
        }
    }
}
```

2．添加学生信息（AddStudentInfo.aspx）

新建一个网页，命名为 AddStudentInfo.aspx，主要用于添加学生信息。该页面中用到的主要控件如表 25.16 所示。

表 25.16　添加学生信息页面中用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　途
TextBox	txtNum	无	输入学生编号
	txtName	无	输入学生名称
TextBox	txtPwd	无	输入新密码
Button	btnSubmit	Text 属性设置为“添加”	添加
	btnConcel	Text 属性设置为“重置”	重置
RadioButtonList	rblSex	Items 属性中添加两项	选择学生性别

确认输入的学生信息无误后，单击“添加”按钮，即可将学生信息添加到存储学生信息的数据表中。代码如下：

```
protected void btnSubmit_Click(object sender, EventArgs e)
```



```
{
    if (txtName.Text == "" || txtNum.Text == "" || txtPwd.Text == "")          //检查信息输入是否完整
    {
        MessageBox.Show("请将信息填写完整");                                     //弹出提示信息
        return;
    }
    else
    {
        SqlConnection conn = BaseClass.DBCon();                                  //连接数据库
        conn.Open();                                                             //打开连接
   SqlCommand cmd = new SqlCommand("select count(*) from tb_Student where StudentNum='" + txtNuxt +
"'", conn);
        int i = Convert.ToInt32(cmd.ExecuteScalar());                            //获取返回值
        if (i > 0)                                                               //如果返回值大于 0
        {
            MessageBox.Show("此学号已经存在");                                   //提示学号已经存在
            return;
        }
        else
        {
             //将新增学生信息添加到数据库中
            cmd = new SqlCommand("insert into tb_Student(StudentNum,StudentName,StudentSex,
StudentPwd) values('" + txtNum.Text.Trim() + "','" + txtName.Text.Trim() + "','" + rblSex.SelectedValue.ToString()
+ "','" + txtPwd.Text. Trim() + "')", conn);
            cmd.ExecuteNonQuery();
            conn.Close();                                                        //关闭连接
            MessageBox.Show("添加成功");                                         //提示添加成功
            btnConcel_Click(sender, e);
        }
    }
}
```

3．管理教师基本信息（TeacherInfo.aspx）

新建一个网页，命名为 TeacherInfo.aspx，主要用于浏览、删除和更改教师信息。此页只需要一个 GridView 控件，这里不进行具体介绍，只给出关键代码。

当加载 TeacherInfo.aspx 页面时，需要对 GridView 控件进行绑定，显示所有的教师信息。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] == null)                                                //禁止匿名登录
    {
        Response.Redirect("../Login.aspx");
    }
    if (!IsPostBack)
```

```
    {
        string strsql = "select * from tb_Teacher order by ID desc";          //检索出所有教师信息
        BaseClass.BindDG(gvTeacher,"ID",strsql,"teacher");                     //绑定控件
    }
}
```

当单击某位教师的编号时，会转向教师详细信息页面（TeacherXXinfo.aspx），在此可以浏览教师的详细信息以及对教师信息进行修改。实现步骤如下。

（1）新建一个网页，命名为 TeacherXXinfo.aspx，主要用于查看教师的详细信息及对教师信息进行修改。该页面中用到的主要控件如表 25.17 所示。

表 25.17　教师详细信息页面中用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　　途
TextBox	txtTNum	无	显示教师编号
	txtTName	无	输入/显示教师姓名
	txtTPwd	无	输入/显示教师登录密码
Button	btnSave	Text 属性设置为“保存”	保存修改
	btnConcel	Text 属性设置为“取消”	取消
RadioButtonList	ddlTKm	无	选择教师负责科目

（2）当此页面加载时，程序会以教师的编号作为查询条件，从数据库中检索出教师的其他信息并显示出来。代码如下：

```
private static int id;                                                          //建立公共变量
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] == null)                                               //禁止匿名登录
    {
        Response.Redirect("../Login.aspx");
    }
    if (!IsPostBack)
    {
        id = Convert.ToInt32(Request.QueryString["Tid"]);                       //获取教师的系统编号
        SqlConnection conn = BaseClass.DBCon();                                 //连接数据库
        conn.Open();                                                            //打开数据库
        SqlCommand cmd = new SqlCommand("select * from tb_Teacher where ID=" + id, conn);
        SqlDataReader sdr = cmd.ExecuteReader();
        sdr.Read();
        txtTName.Text = sdr["TeacherName"].ToString();                          //显示教师姓名
        txtTNum.Text = sdr["TeacherNum"].ToString();                            //显示教师登录账号
        txtTPwd.Text = sdr["TeacherPwd"].ToString();                            //显示教师登录密码
        int kmid = Convert.ToInt32(sdr["TeacherCourse"].ToString());            //获取教师授课科目编号
        sdr.Close();
        cmd = new SqlCommand("select LessonName from tb_Lesson where ID=" + kmid, conn);
        string KmName = cmd.ExecuteScalar().ToString();                         //显示科目名称
```

```
            cmd = new SqlCommand("select * from tb_Lesson", conn);
            sdr = cmd.ExecuteReader();
            ddlTKm.DataSource = sdr;                                  //设置数据源
            ddlTKm.DataTextField = "LessonName";                      //设置显示字段名称
            ddlTKm.DataValueField = "ID";
            ddlTKm.DataBind();
            ddlTKm.SelectedValue =kmid.ToString();
            conn.Close();
      }
}
```

（3）如果想修改教师信息，更改教师现有信息后，单击“保存”按钮对教师信息进行修改。代码如下：

```
protected void btnSava_Click(object sender, EventArgs e)
{
      if (txtTName.Text == "" || txtTPwd.Text == "")                  //检查信息是否输入完整
      {
            MessageBox.Show("请将信息填写完整");                        //弹出提示信息
            return;
      }
      else
      {
            string strsql="update tb_Teacher set TeacherName='" + txtTName.Text.Trim() + "',TeacherPwd='" +
txtTPwext. Trim() + "',TeacherCourse='"+ddlTKm.SelectedValue.ToString()+"' where ID="+id;
            BaseClass.OperateData(strsql);                            //执行更新教师信息表
            Response.Redirect("TeacherInfo.aspx");                    //转向教师基本信息
      }
}
```

4．添加教师信息（AddTeacherInfo.aspx）

新建一个网页，命名为 AddTeacherInfo.aspx，主要用于添加教师的详细信息。该页面中用到的主要控件如表 25.18 所示。

表 25.18　添加教师信息页面中用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　途
abl TextBox	txtTeacherNum	无	输入教师编号
	txtTeacherName	无	输入教师姓名
	txtTeacherPwd	无	输入教师登录密码
ab Button	btnAdd	Text 属性设置为“添加”	添加
	btnconcel	Text 属性设置为“重置”	重置
RadioButtonList	ddlTeacherKm	无	选择教师负责科目

确认输入的教师信息无误后，单击“添加”按钮即可将新增教师信息添加到数据表中。代码如下：

```
protected void btnAdd_Click(object sender, EventArgs e)
```

```
{
    //检查信息输入是否完整
    if (txtTeacherName.Text == "" || txtTeacherNum.Text == "" || txtTeacherPwd.Text == "")
    {
        MessageBox.Show("请将信息填写完整");                              //弹出提示信息
        return;
    }
    else
    {
        SqlConnection conn = BaseClass.DBCon();                           //连接数据库
        conn.Open();                                                      //打开数据库
        SqlCommand cmd = new SqlCommand("select count(*) from tb_Teacher where Teachm=
'"+txtTeacherNum. Text. Trim()+"'", conn);
        int t = Convert.ToInt32(cmd.ExecuteScalar());                     //获取返回值
        if (t > 0)                                                        //判断返回值是否大于 0
        {
            MessageBox.Show("此教师编号已经存在");                        //弹出提示信息
            return;
        }
        else
        {
            //将信息添加到数据库中
            string str = "insert into tb_Teacher(TeacherNum,TeacherName,TeacherPwd,TeacherCourse)
values('" + txtTerNum.Text.Trim() + "','" + txtTeacherName.Text.Trim() + "','" + txtTeacherPwd.Text.Trim() + "','"
+ ddlTeKm. SelectedValue.ToString() + "')";
            BaseClass.OperateData(str);
            MessageBox.Show("教师信息添加成功");                          //提示信息添加成功
            btnconcel_Click(sender, e);
        }
    }
}
```

5．试题基本信息管理（ExaminationInfo.aspx）

新建一个网页，命名为 ExaminationInfo.aspx，主要用于查看试题详细信息、查询试题以及对试题进行删除和修改。该页面中用到的主要控件如表 25.19 所示。

表 25.19　试题基本信息管理页面中用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　　途
Button	btnSerch	Text 属性设置为“查看”	查询
GridView	gvExaminationInfo	Columns 属性中添加 4 列	显示试题题目信息及对试题的各项操作
DropDownList	ddlEkm	无	选择查询范围

ExaminationInfo.aspx 页面加载时，会将所有的试题信息绑定到 GridView 控件上显示出来，并且将所有的科目名称绑定到 DropDownList 控件上。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] == null)                                          //禁止匿名登录
    {
        Response.Redirect("../Login.aspx");
    }
    if (!IsPostBack)
    {
        string strsql = "select * from tb_test order by ID desc";          //检索所有试题信息
        BaseClass.BindDG(gvExaminationInfo, "ID", strsql, "ExaminationInfo");   //绑定控件
        SqlConnection conn = BaseClass.DBCon();                            //连接数据库
        conn.Open();                                                       //打开数据库
        SqlCommand cmd = new SqlCommand("select * from tb_Lesson", conn);
        SqlDataReader sdr = cmd.ExecuteReader();
        this.ddlEkm.DataSource = sdr;                                      //设置数据源
        this.ddlEkm.DataTextField = "LessonName";                          //设置显示字段
        this.ddlEkm.DataValueField = "ID";
        this.ddlEkm.DataBind();
        this.ddlEkm.SelectedIndex = 0;
        conn.Close();                                                      //关闭连接
    }
}
```

单击每条试题信息的“详细信息”按钮，将弹出显示试题详细信息页面。实现显示试题详细信息页面的方法如下。

（1）新建一个网页，命名为 ExaminationDetail.aspx，主要用于显示试题的详细信息以及更改试题信息。该页面中用到的主要控件如表 25.20 所示。

表 25.20　显示试题详细信息页面中用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　途
TextBox	txtsubject	TextMode 属性设置为 MultiLine	输入/显示试题题目
	txtAnsA	TextMode 属性设置为 MultiLine	输入/显示答案选项 A
	txtAnsB	TextMode 属性设置为 MultiLine	输入/显示答案选项 B
	txtAnsC	TextMode 属性设置为 MultiLine	输入/显示答案选项 C
	txtAnsD	TextMode 属性设置为 MultiLine	输入/显示答案选项 D
Button	btnconfirm	Text 属性设置为“确定”	确定
	btnconcel	Text 属性设置为“取消”	取消
RadioButtonList	rblRightAns	Items 属性中添加 4 项	显示/选择正确答案
CheckBox	cbFB	Text 属性设置为“是否发布”	显示/设置是否发布
Label	lblkm	无	显示教师负责的课程

（2）ExaminationDetail.aspx 页面加载时，程序根据试题的系统编号 id 查询出试题的其他信息并显示出来。关键代码如下：

```
private static int id;
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] == null)                                          //禁止匿名登录
    {
        Response.Redirect("../Login.aspx");
    }
    if (!IsPostBack)
    {
        id = Convert.ToInt32(Request.QueryString["Eid"]);                  //获取试题的系统编号
        SqlConnection conn = BaseClass.DBCon();                            //连接数据库
        conn.Open();                                                       //打开连接
        SqlCommand cmd = new SqlCommand("select * from tb_test where ID="+id, conn);
        SqlDataReader sdr = cmd.ExecuteReader();
        sdr.Read();
        txtsubject.Text = sdr["testContent"].ToString();                   //显示试题题目
        txtAnsA.Text = sdr["testAns1"].ToString();                         //显示试题选项 A
        txtAnsB.Text = sdr["testAns2"].ToString();                         //显示试题选项 B
        txtAnsC.Text = sdr["testAns3"].ToString();                         //显示试题选项 C
        txtAnsD.Text = sdr["testAns4"].ToString();                         //显示试题选项 D
        rblRightAns.SelectedValue = sdr["rightAns"].ToString();            //显示正确答案
        string fb = sdr["pub"].ToString();                                 //获取是否发布
        if (fb == "1")
            cbFB.Checked = true;
        else
            cbFB.Checked = false;
        lblkm.Text = sdr["testCourse"].ToString();                         //显示试题所属科目
        sdr.Close();
        conn.Close();                                                      //关闭连接
    }
}
```

（3）如果想修改试题信息，在确认输入的修改信息无误后，单击“确定”按钮完成对试题信息的修改。代码如下：

```
protected void btnconfirm_Click(object sender, EventArgs e)
{
    //检查输入信息是否完整
    if (txtsubject.Text == "" || txtAnsA.Text == "" || txtAnsB.Text == "" || txtAnsC.Text == "" || txtAnsD.Text == "" )
    {
        MessageBox.Show("请将信息填写完整");                                  //弹出提示信息
        return;
    }
    else
    {
        string isfb = "";
```

```
            if (cbFB.Checked == true)                              //判断是否选中
                isfb = "1";
            else
                isfb = "0";
                                                                   //更新数据库中试题信息表
            string str="update tb_test set testContent='" + txtsubject.Text.Trim() + "',testAns1='" +
txtAnsA.Text.Trim() + "', tens2='" + txtAnsB.Text.Trim() + "',testAns3='" + txtAnsC.Text.Trim() + "',testAns4='" +
txtAnsD.Text + "', rightAns='" + rblRtAns. SelectedValue.ToString() + "',pub='"+isfb+"' where ID=" + id;
            BaseClass.OperateData(str);                            //执行 SQL 语句
            Response.Redirect("ExaminationInfo.aspx");
        }
 }
```

6．添加试题信息（AddExamination.aspx）

新建一个网页，命名为 AddExamination.aspx，主要用于添加试题信息，由于该页面中用到的控件与显示试题详细信息页面中所需的控件基本相同，所以此处不进行详细介绍，只给出关键代码。

确认输入的新增试题信息无误后，单击“确定”按钮将试题信息添加到试题信息表中。代码如下：

```
protected void btnconfirm_Click(object sender, EventArgs e)
{
                                                                   //检查输入信息是否完整
    if (txtsubject.Text == "" || txtAnsA.Text == "" || txtAnsB.Text == "" || txtAnsC.Text == "" || txtAnsD.Text == "")
    {
        MessageBox.Show("请将信息填写完整");                       //弹出提示信息
        return;
    }
    else
    {
        string isfb = "";
        if (cbFB.Checked == true)                                  //判断是否选中
            isfb = "1";
        else
            isfb = "0";
        //将信息插入数据库中的试题信息表中
        string str = "insert into tb_test(testContent,testAns1,testAns2,testAns3,testAns4,rightAns,pub,testCourse)
values ('" + txtsubject. Text.Trim() + "','" + txtAnsA.Text.Trim() + "','" + txtAnsB.Text.Trim() + "','" +
txtAnsC.Text.Trim() + "','" + txtAnsD.Text.Trim() + "','" + rblRightAns.SelectedValue.ToString() + "','" + isfb + "','"
+ ddlkm.SelectedItem.Text + "')";
        BaseClass.OperateData(str);                                //执行 SQL 语句
        btnconcel_Click(sender,e);
    }
}
```

7．考试科目设置（Subject.aspx）

新建一个网页，命名为 Subject.aspx，主要用于显示、添加和删除考试科目信息。该页面中用到的

主要控件如表 25.21 所示。

表 25.21　考试科目设置页面中用到的主要控件

控件类型	控件 ID	主要属性设置	用　途
Button	btnAdd	Text 属性设置为“添加”	添加
	btnDelete	Text 属性设置为“删除”	删除
TextBox	txtKCName	无	输入新增科目名称
ListBox	ListBox1	无	显示所有科目

页面加载时，程序将所有的科目信息检索出来显示在 ListBox 控件上。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] == null)                                   //禁止匿名登录
    {
        Response.Redirect("../Login.aspx");
    }
    if (!IsPostBack)
    {
        SqlConnection conn = BaseClass.DBCon();                      //连接数据库
        conn.Open();                                                 //打开连接
        SqlCommand cmd = new SqlCommand("select * from tb_Lesson", conn);
        SqlDataReader sdr = cmd.ExecuteReader();
        while (sdr.Read())
        {
            ListBox1.Items.Add(sdr["LessonName"].ToString());        //为 ListBox 添加项
        }
    }
}
```

输入新增科目信息后，单击“添加”按钮将信息添加到考试科目信息表（tb_Lesson）中。代码如下：

```
protected void btnAdd_Click(object sender, EventArgs e)
{
     if (txtKCName.Text == "")                                       //判断是否输入课程名称
     {
         MessageBox.Show("请输入课程名称");                            //弹出提示信息
         return;
     }
     else
     {
         string systemTime = DateTime.Now.ToString();                //获取当前系统时间
         //将信息插入数据库的课程信息表中
         string strsql = "insert into tb_Lesson(LessonName,LessonDataTime) values('" + txtKCName.
Text.Trim() + "','" + smTime + "')";
         BaseClass.OperateData(strsql);                              //执行 SQL 语句
```

```
            txtKCName.Text = "";
            Response.Write("<script>alert('添加成功');location='Subject.aspx'</script>");
        }
}
```

在 ListBox 控件中选择要删除的科目，单击“删除”按钮将科目删除。代码如下：

```
protected void btnDelete_Click(object sender, EventArgs e)
{
    if (ListBox1.SelectedValue.ToString() == "")                    //判断是否有选中项
    {
        MessageBox.Show("请选择删除项目后删除");                      //弹出提示
        return;
    }
    else
    {
        //删除指定的信息
        string strsql = "delete from tb_Lesson where LessonName='" + ListBox1.SelectedItem.Text + "'";
        BaseClass.OperateData(strsql);                              //执行 SQL 语句
        Response.Write("<script>alert('删除成功');location='Subject.aspx'</script>");
    }
}
```

8．查询考试结果（ExaminationResult.aspx）

新建一个网页，命名为 ExaminationResult.aspx，主要用于显示考试记录信息，该页面中只使用了 GridView 控件，此处不进行详细介绍，只给出关键代码。

此页面加载时，程序将所有考试记录检索出来显示在 GridView 控件上。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] == null)                                   //禁止匿名登录
    {
        Response.Redirect("../Login.aspx");
    }
    if (!IsPostBack)
    {
        string strsql = "select * from tb_score order by ID desc";  //检索所有考试结果信息
        BaseClass.BindDG(gvExaminationresult,"ID",strsql,"result"); //绑定控件
    }
}
```

如果想删除某条信息，可以单击与信息对应的“删除”按钮。代码如下：

```
protected void gvExaminationInfo_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    int id = (int)gvExaminationresult.DataKeys[e.RowIndex].Value;   //获取欲删除的信息编号
    string strsql = "delete from tb_score where ID=" + id;          //删除指定编号的信息
```

```
    BaseClass.OperateData(strsql);                                    //执行 SQL 语句
    string strsql1 = "select * from tb_score order by ID desc";       //检索所有考试结果信息
    BaseClass.BindDG(gvExaminationresult, "ID", strsql1, "result");   //绑定控件
}
```

9．管理员信息维护（AdminChangePwd.aspx）

新建一个网页，命名为 AdminChangePwd.aspx，主要用于管理员修改密码。该页面中用到的主要控件如表 25.22 所示。

表 25.22　管理员信息维护页面中用到的主要控件

控 件 类 型	控件 ID	主要属性设置	用　　途
TextBox	txtOldPwd	无	输入旧密码
	txtNewPwd	无	输入新密码
	txtNewPwdA	无	再输入一次新密码
Button	btnchange	Text 属性设置为“确定修改”	确定修改

如果要更改管理员密码，系统首先要求输入旧密码，然后再输入新密码，如果旧密码输入错误，系统会弹出提示框。代码如下：

```
protected void btnchange_Click(object sender, EventArgs e)
{
    //检查输入信息是否完整
    if (txtNewPwd.Text == "" || txtNewPwdA.Text == "" || txtOldPwd.Text == "")
    {
        MessageBox.Show("请将信息填写完整");                               //弹出提示信息
        return;
    }
    else
    {
        if (BaseClass.CheckAdmin(Session["admin"].ToString(), txtOldPwd.Text.Trim())) //验证旧密码是否正确
        {
            if (txtNewPwd.Text.Trim() != txtNewPwdA.Text.Trim())          //检查两次输入是否一致
            {
                MessageBox.Show("两次密码不一致");                           //弹出提示信息
                return;
            }
            else
            {
                string strsql = "update tb_Admin set AdminPwd='" + txtNewPwdA.Text.Trim() + "' where
Admin='"+Session["admin"].ToString()+"'";                                   //更新数据库中的管理员信息表
                BaseClass.OperateData(strsql);                             //执行 SQL 语句
                MessageBox.Show("密码修改成功");
                txtNewPwd.Text = "";
                txtNewPwdA.Text = "";
                txtOldPwd.Text = "";
            }
```

```
            }
            else
            {
                MessageBox.Show("旧密码输入错误");
                return;
            }
        }
}
```

25.10　小　结

通过开发在线考试系统，总结出在线考试系统最基本的是要具备登录、随机抽取试题、答卷和评分。可以说这 4 部分组成了在线考试系统，而其他一些功能或者模块都是间接地服务于这 4 部分。当然，完善的在线考试系统，也要具备优良的后台管理模块，只有将后台管理模块设计完善，才能使整个系统变得更加灵活和容易维护。读者只要能够理解本章涉及的知识点，便可自行开发出一套完善的在线考试系统。